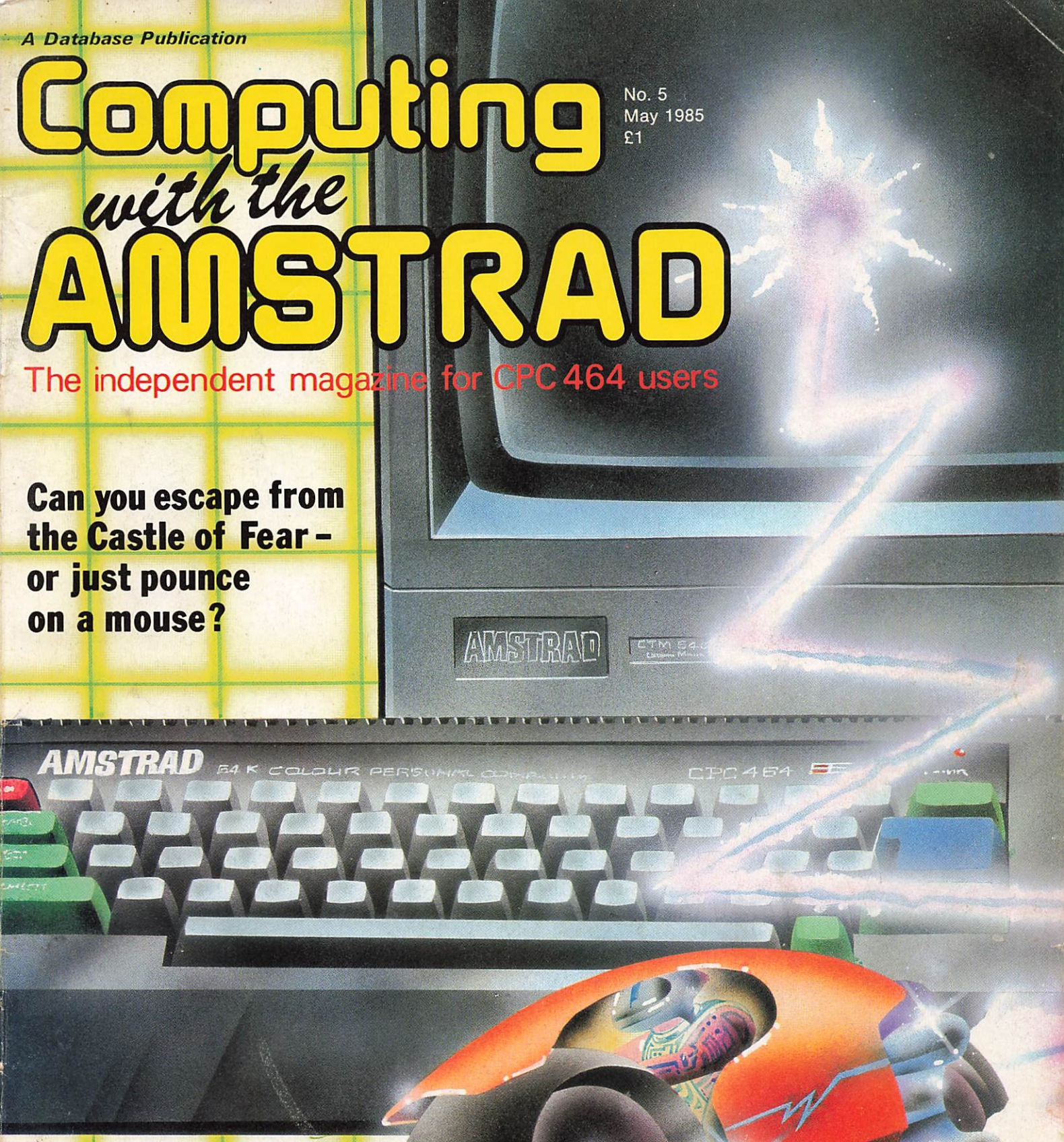
A Database Publication

Computing with the AMSTRAD

No. 5
May 1985
£1

The independent magazine for CPC 464 users

Can you escape from the Castle of Fear – or just pounce on a mouse?



Amstrad teach-in

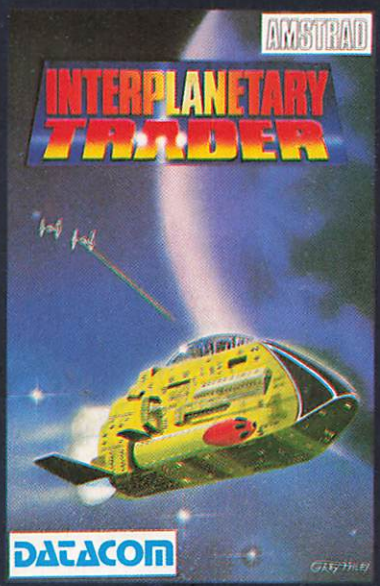
Learn about loops and INPUT . . . experiment with Z80 registers . . . extra commands for Basic . . . animation by palette switching . . . and create your own character set

Duel to the death in... TRON CYCLES

WELCOME TO THE WORLD OF DATACOM

EXECUTION - They told me it would be bad but I never thought it would be like this...must keep a clear mind... can't afford to panic...time is fast running out...don't think my nerves will stand much more of this!!! A brain straining memory bashing game of words.

SNAIL PACE - Unbearable excitement. 30 'thoroughbred racing snails' battle to the finish. Super features include computer calculated odds, form guide, excellent sound effects, smooth m/c graphics. Rivetting excitement guaranteed.



INTERPLANETARY TRADER - The most addictive, mind blowing, feature packed space adventure you will ever experience on your AMSTRAD CPC 464. Defend your cargo against space pirates...navigate asteroids, black holes, magnetic storms...Total concentration is required in your quest to become a GALACTIC MEGABILLIONAIRE!!!

DATACOM PUBLICATIONS:
407F Hockley Centre,
Birmingham B18 6NF,
Tel: 021-233 1800

Trade enquiries welcome

All games just

£5.95

Each

These games will be available from your dealer or **POST FREE*** direct from: DATACOM PUBLICATIONS using the coupon provided (or just write in)

* U.K. only - Overseas add £1 p & p

Name .....

Address .....

Please rush me ☐ Tapes of Interplanetary Trader ☐ Tapes of Execution ☐ Tapes of Snail Pace

I enclose a cheque/P.O. for £.....

Tel: .....

Vol. 1 No. 5 May 1985

Managing Editor: **Derek Meakin**
Features Editor: **Peter Bibby**
The A Team: **Mike Bibby**
Alan McLachlan
Kevin Edwards
Roland Waddilove
Production Editor: **Peter Glover**
Layout Design: **Heather Sheldrick**
News editor: **Mike Cowley**
Advertisement Manager: **John Riding**
Advertising Sales: **Margaret Clarke**
Editor in Chief: **Peter Brameld**

Editorial: 061-456 8835
Administration: 061-456 8383
Advertising: 061-456 8500
Subscriptions: 061-480 0171
Telex: 667664 SHARET G
Prestel Mailbox: 614568383

Published by:
Database Publications Ltd,
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.

Subscription rates for 12 issues, post free:
£12 – UK
£15 – Eire (Sterling only)
£20 – Rest of world (surface)
£40 – Rest of world (airmail)

ABC Member of Audit Bureau of Circulations

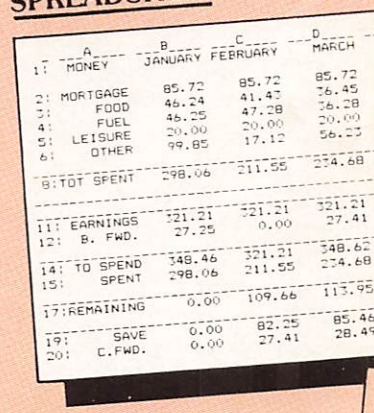
Contents

QUICK TO LEAR

THAT'S...

MINI OFFICE

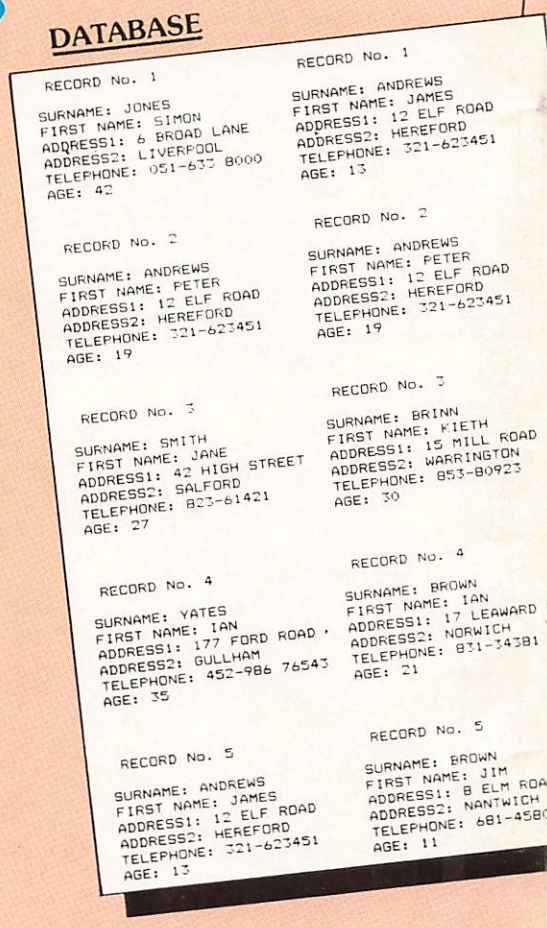
SPREADSHEET



	A MONEY	B JANUARY	C FEBRUARY	D MARCH
2:	MORTGAGE	85.72	85.72	85.72
3:	FOOD	46.24	41.43	36.45
4:	FUEL	46.25	47.28	36.28
5:	LEISURE	20.00	20.00	20.00
6:	OTHER	99.85	17.12	56.23
8:	TOT SPENT	298.06	211.55	234.68
11:	EARNINGS	321.21	321.21	321.21
12:	B. FWD.	27.25	0.00	27.41
14:	TO SPEND	348.46	321.21	348.62
15:	SPENT	298.06	211.55	234.68
17:	REMAINING	0.00	109.66	113.95
19:	SAVE	0.00	82.25	85.46
20:	C.FWD.	0.00	27.41	28.49

JUST LOOK WHAT THIS PACKAGE CAN DO!

DATABASE



RECORD No. 1
SURNAME: JONES
FIRST NAME: SIMON
ADDRESS1: 6 BROAD LANE
ADDRESS2: LIVERPOOL
TELEPHONE: 051-633 8000
AGE: 42

RECORD No. 1
SURNAME: ANDREWS
FIRST NAME: JAMES
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 13

RECORD No. 2
SURNAME: ANDREWS
FIRST NAME: PETER
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 19

RECORD No. 2
SURNAME: ANDREWS
FIRST NAME: PETER
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 19

RECORD No. 3
SURNAME: SMITH
FIRST NAME: JANE
ADDRESS1: 42 HIGH STREET
ADDRESS2: SALFORD
TELEPHONE: 823-61421
AGE: 27

RECORD No. 3
SURNAME: BRINN
FIRST NAME: KIETH
ADDRESS1: 15 MILL ROAD
ADDRESS2: WARRINGTON
TELEPHONE: 853-80923
AGE: 30

RECORD No. 4
SURNAME: YATES
FIRST NAME: IAN
ADDRESS1: 177 FORD ROAD
ADDRESS2: GULLHAM
TELEPHONE: 452-986 76543
AGE: 35

RECORD No. 4
SURNAME: BROWN
FIRST NAME: IAN
ADDRESS1: 17 LEAWARD
ADDRESS2: NORWICH
TELEPHONE: 831-34381
AGE: 21

RECORD No. 5
SURNAME: ANDREWS
FIRST NAME: JAMES
ADDRESS1: 12 ELF ROAD
ADDRESS2: HEREFORD
TELEPHONE: 321-623451
AGE: 13

RECORD No. 5
SURNAME: BROWN
FIRST NAME: JIM
ADDRESS1: 8 ELM ROA
ADDRESS2: NANTWICH
TELEPHONE: 681-4580
AGE: 11

WORD PROCESSOR – Ideal for writing letters or reports! *Features:* Constant time display ● Constant word count (even shows words per minute) ● Normal or double-height text on screen or printout.

SPREADSHEET – Use your micro to manage your money! *Features:* Number display in rows and columns ● Continuous updating ● Update instantly reflected throughout spreadsheet ● Save results for future amendments.

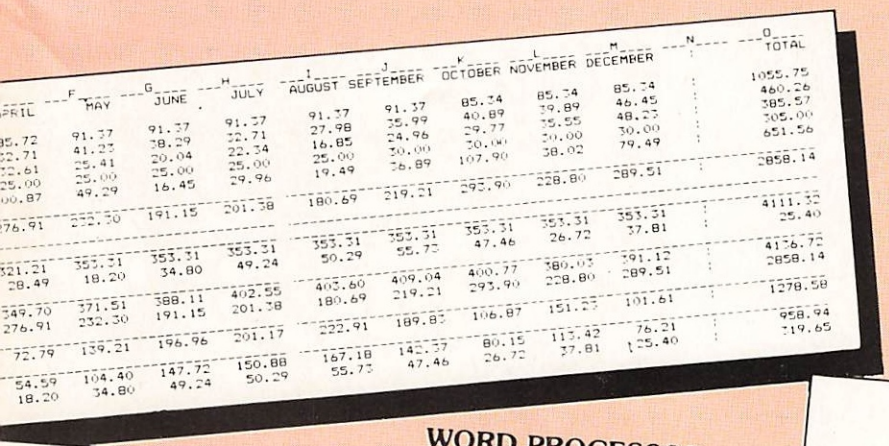
GRAPHICS – Turn those numbers into an exciting visual display! *Features:* 3D bar chart ● Pie chart ● Graph.

DATABASE – Use it like an office filing cabinet! *Features:* Retrieve files at a keystroke ● Sort ● Replace ● Save ● Print ● Search.

...and it's all at price of just

N, EASY TO USE

Specially written for your AMSTRAD CPC 464



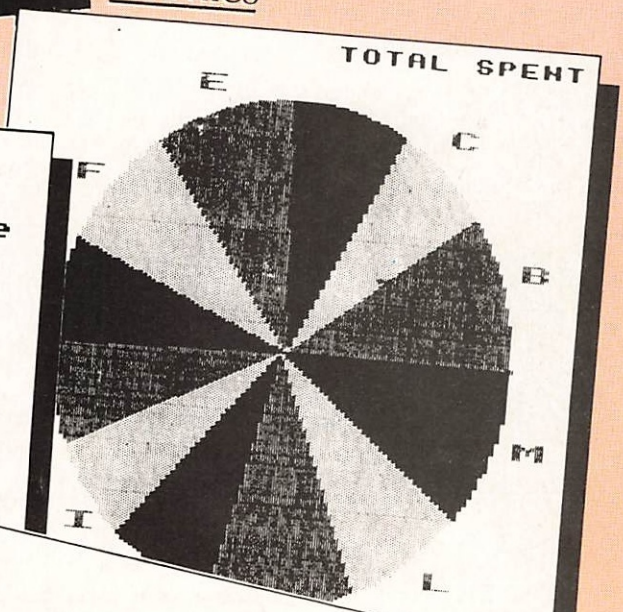
APRIL	F MAY	G JUNE	H JULY	I AUGUST	J SEPTEMBER	K OCTOBER	L NOVEMBER	M DECEMBER	N	O TOTAL
85.72	91.37	91.37	91.37	91.37	91.37	85.34	85.34	85.34		1055.75
22.71	41.23	38.29	32.71	27.98	35.99	40.89	39.89	46.45		460.26
72.61	25.41	20.04	22.34	16.85	24.96	29.77	35.55	48.23		385.57
25.00	25.00	25.00	25.00	25.00	30.00	30.00	30.00	30.00		305.00
00.87	49.29	16.45	29.96	19.49	36.89	107.90	38.02	79.49		651.56
276.91	232.30	191.15	201.38	180.69	219.21	293.90	228.80	289.51		2858.14
321.21	353.31	353.31	353.31	353.31	353.31	353.31	353.31	353.31		4111.32
28.49	18.20	34.80	49.24	50.29	55.73	47.46	26.72	37.81		25.40
349.70	371.51	388.11	402.55	403.60	409.04	400.77	380.03	391.12		4136.72
276.91	232.30	191.15	201.38	180.69	219.21	293.90	228.80	289.51		2858.14
	72.79	139.21	196.96	201.17	222.91	189.83	106.87	151.23	101.61	1270.58
	54.59	104.40	147.72	150.88	167.18	142.37	80.15	113.42	76.21	958.94
	18.20	34.80	49.24	50.29	55.73	47.46	26.72	37.81	25.40	719.65

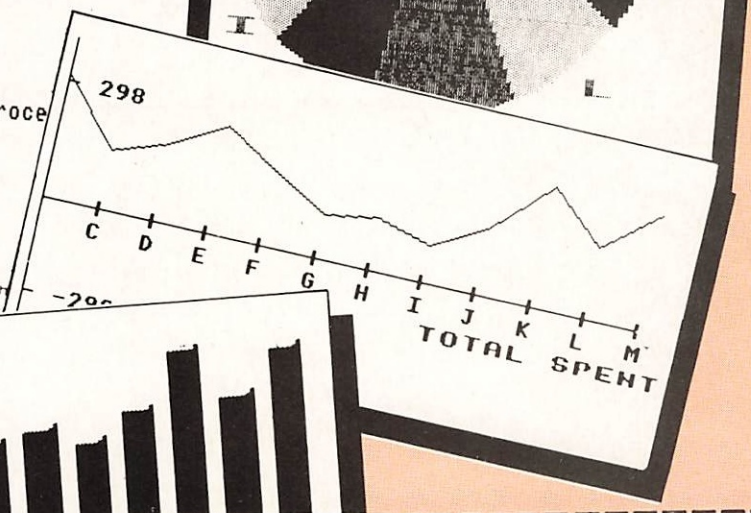
WORD PROCESSOR

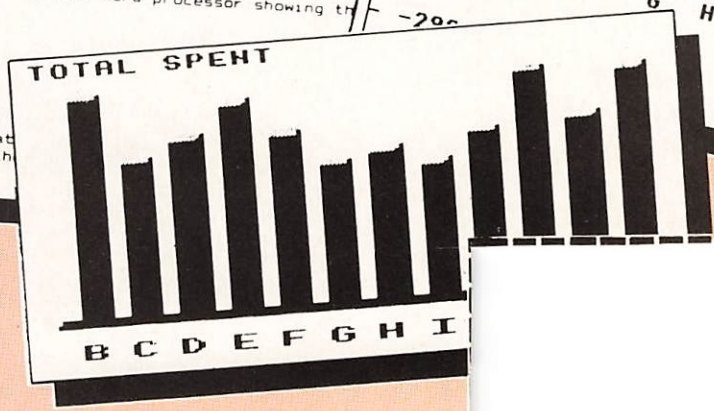
Page 1

This is a demonstration of the
MINI OFFICE word processor
showing the various printout
options available.

GRAPHICS



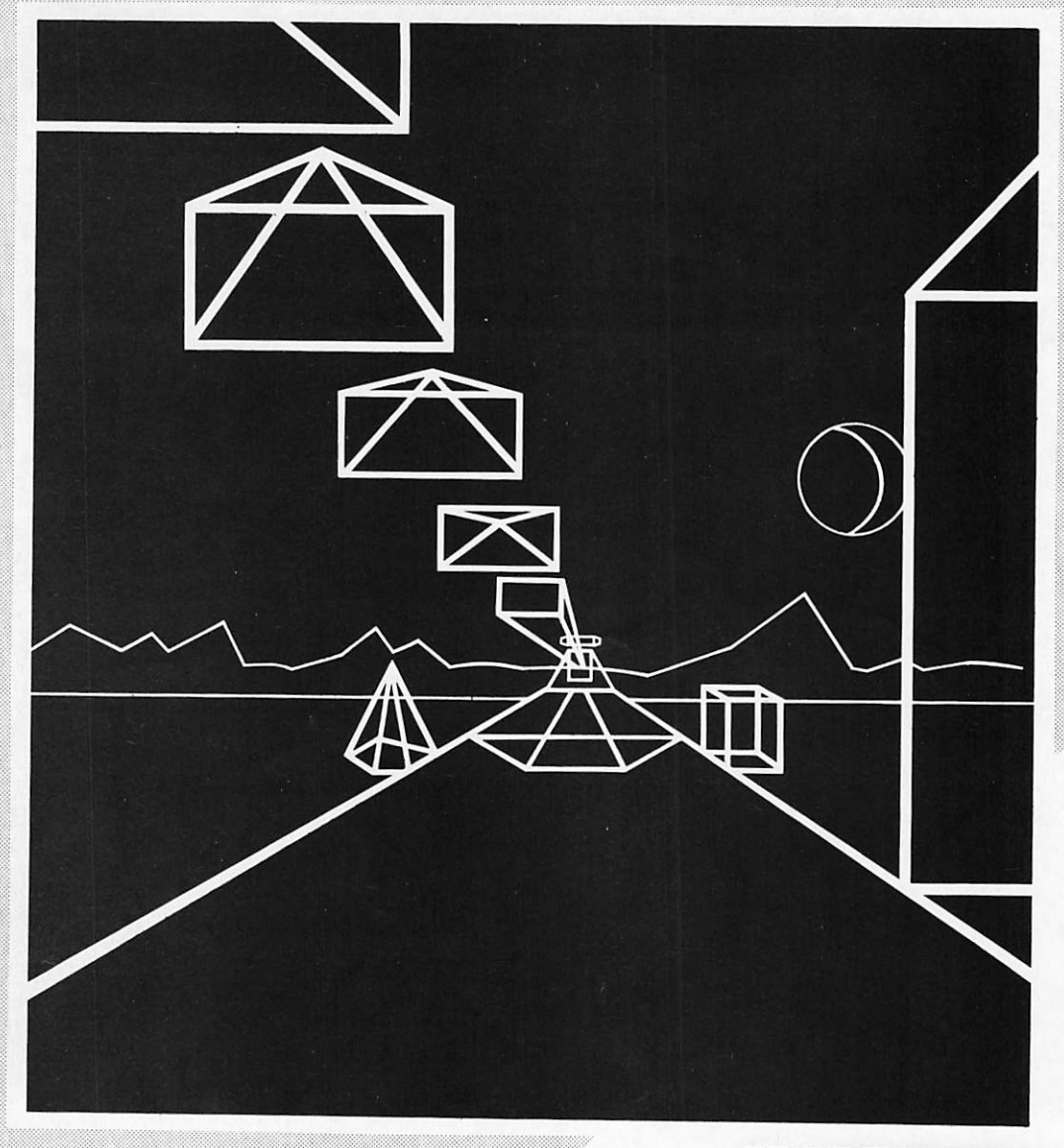




the unbelievable

£5.95
CASSETTE
3" DISC £9.95

DATABASE SOFTWARE



TANK BUSTERS

Using the advanced graphics techniques developed for the game "Dark Star" the Design Design team bring you TANK BUSTERS. You take control of an advanced battletank on the battlefield of the future.
Your mission is to seek and destroy the enemy forces intent on your annihilation.
R.R.P. £7.95

DARK STAR

At last available on the Amstrad, the game that was awarded 10 out of 10 in Personal Computer News and the only game ever to receive a 100% rating in Crash Magazine. This is the fastest three dimensional space simulation available for any home computer.
R.R.P. £7.95

Design Design

Design-Design Software,
2 Ashton Way, East Herrington,
Sunderland SR3 3RX.

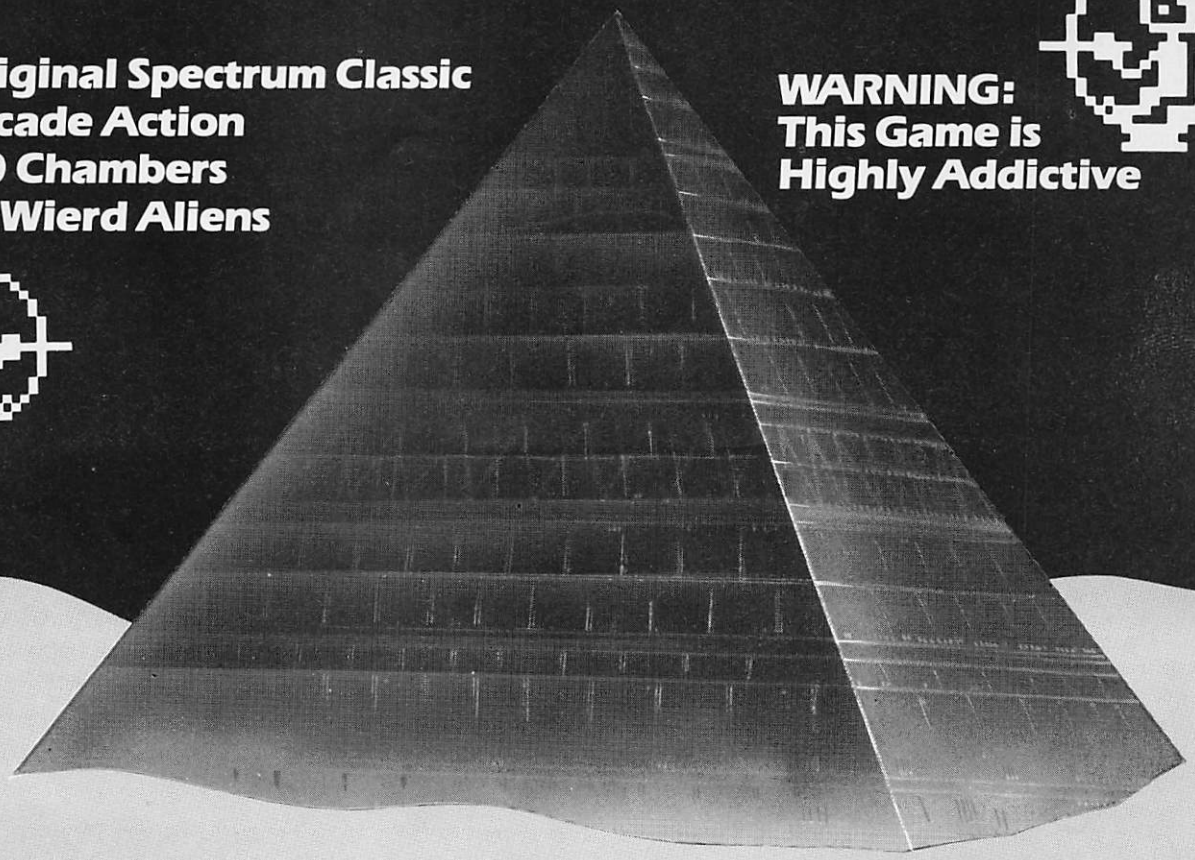
Trade enquiries to:—

125 Smedley Rd., Cheetham Hill,
Manchester M8 7RS
Telephone 061 205 6603

fantasy
SOFTWARE

- Original Spectrum Classic
- Arcade Action
- 120 Chambers
- 60 Wierd Aliens

WARNING:
This Game is Highly Addictive



THE PYRAMID

NOW AVAILABLE FOR
AMSTRAD
£4.95
inc p&p

THE PYRAMID IS AVAILABLE FROM ALL GOOD SOFTWARE RETAILERS OR FROM FANTASY SOFTWARE, 27A ST GEORGES ROAD, CHELTENHAM, GLOUCESTERSHIRE AT **£4.95** INCLUDING POST AND PACKING

Let your fingers do the talking...with this special

Now you can teach your Amstrad to talk!

How it works

AT the heart of the dk'tronics speech synthesiser lies an incredibly powerful chip that has split the English language into its component parts – or allophones as they are known.

Altogether there are 59 allophones and five pauses stored in the speech chip's internal ROM. These can be combined to create a virtually unlimited vocabulary.

The potential of this chip is realised by dk'tronic's sophisticated, yet simple to use software. The brilliant program design enables the Amstrad to actually speak the words you type, in straightforward English, without having to resort to complicated phonetic spelling or difficult programming techniques.

Written to be as user friendly as possible, the synthesiser adds eight powerful commands to Amstrad Basic.

If you prefer complete control over your programs, though, full details are given for Basic and machine code programmers to exploit the tremendous scope of the synthesiser without using the software supplied.

In fact this system supports four different modes of use.

The first mode allows you to sound words using only the Amstrad's normal Basic commands. However, as you get more ambitious with your speech, a second mode is provided. This gives eight extra commands to use from Basic, making using the synthesiser even easier.

The third mode is the text to speech converter. When this is in operation speech can be typed in using normal English and the Amstrad does the rest. There's no need to work out the allophones as in the other two modes – the Amstrad does it for you.

As if all this wasn't enough there's the fourth mode. This has the synthesiser converting whatever appears on the screen into speech. Using this, you can literally listen to your listings!

YOU can add an exciting new dimension to computing with your Amstrad – with the help of this remarkable new product from dk'tronics.

It comes complete with the latest and very versatile speech chip, a powerful stereo amplifier and two high-quality 4in speakers, specially designed to match the Amstrad CPC464.

And because this is a special reader offer it comes to you at £5 off the normal retail price of of £39.95!

Fitting it is simplicity itself. All you have to do is to plug the synthesiser's interface into the floppy disc port at the back of the Amstrad and the jack plug into the stereo socket – and away you go!

With its volume and balance controls you will find you can put dramatic realism into the sound output of your Amstrad. All sounds that previously came from the Amstrad's 1½in mono speakers are now sent out via the interface in stereo.

So even when you're not using it as a speech synthesiser, it can bring startling depth and drama to the music and sound effects of all your favourite games!

These are the sounds – and pauses – you can create on your Amstrad

A	AE	26	fat	F	FF	40	fire	NG	NG	44	bans	TH	TH	29	thin
A	AY	20	great	G	GG2	61	go	O	O	23	cot	TH	DH1	18	they
AI	AIR	47	hair	IG	GG3	34	wig	O	OW	53	snow	TH	DH2	54	bathe
AR	AR	59	farm	GU	GG1	36	guest	O	UW1	22	do	U	AX	15	succeed
AU	AO	24	aught	GE	ZH	38	beige	OO	UW2	31	food	UH	OO	30	cook
B	BB1	28	rib	H	HH1	27	he	OR	OR	58	store	U	YY1	49	compute
B	BB2	63	big	H	HH2	57	hoe	OU	OW	32	ouch	V	VV	35	even
C	KK1	42	common	I	IH	12	fitting	OY	OY	5	toy	W	WW	46	wool
'C	C	8	uncle	I	Y	6	sky	P	PP	9	pub	WH	WH	48	whig
K	KK2	41	sky	IR	ER2	52	bird	R	RR1	14	real	Y	YY2	25	yes
CH	CH	50	church	J	JH	10	jury	R	RR2	39	brain	Z	ZZ	43	zoo
D	DD1	21	could	L	LL	45	luck	R	YR	60	fear	PA1	PA1	0	10 mS
D	DD2	33	do	L	EL	62	angle	S	SS	55	sat	PA2	PA2	1	30 mS
E	EH	7	bend	M	MM	16	milk	SH	SH	37	shirt	PA3	PA3	2	50 mS
E	EE	19	see	N	NN1	11	earn	T	TT1	17	its	PA4	PA4	3	100 mS
ER	ER	51	cater	N	NN2	56	no	T	TT2	13	top	PA5	PA5	5	200 mS

Column 1: Sound. Column 2: Allophone name. Column 3: Allophone number. Column 4: Example word.

Computing with the Amstrad reader offer!

Look at what this package offers you:

- ★ Speech synthesiser with almost unlimited vocabulary
- ★ Easy-to-use commands – it accepts normal English words
- ★ Built-in stereo amplifier with twin speakers
- ★ Programs can run while the speed chip talks

Eight additional Basic commands

|SPON	Speech on.
|SPOF	Speech off.
|FEED,n	Feed speech buffer direct.
|FLUS	Clear speech and text buffers.
|SPED,n	Speech speed.
|OUTM,1	PRINT text to speech.
|OUTM,2	Screen output to speech.
|OUTM,3	Output to screen and speech.

Please send me the dk'tronics speech synthesiser for my Amstrad CPC464

☐ I enclose cheque for £34.95 *(incl. VAT, p&p)* made payable to Database Publications Ltd.

I wish to pay by

☐ Access Card No. _____

☐ Visa Card No. _____

Signed _____

Name _____

Address _____

_____

POST TO: Speech Synthesiser Offer, Database Publications, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

CA5

Allow 28 days for delivery

AMSTRAD BBC CBM64 SPECTRUM MEMOTECH ENTERPRISE ATARI MSX

It finally happened. Despite saying that I'd never get addicted to adventure games, there I was at two in the morning, staring bleakly at the screen. "What now?" on my television screen. Just how on earth do you get past that hostile woolly mammoth?

The game in question was *Lords of Time* from Level 9 and yes, I was hooked. I didn't need any second telling, therefore, when I was given the chance to meet the creators of the Time Lords, woolly mammoth and all, at their hideout in High Wycombe.

Ask any dedicated adventure player to name a software house, and it's quite likely that the answer will be Level 9. Not only do they produce a range of consistently high class ...

If anyone asked me to recommend a first adventure game to try, I don't think there would be any hesitation in my reply. The obvious choice is Colossal Adventure from Level 9.

In the first place, Colossal Adventure is the original. Scott Adams may have been the first programmer to fit an Adventure onto a lowly home microcomputer (the TRS-80), but his inspiration and the start of the genre was a piece of software written by Americans Crowther and Woods on a mainframe computer back in 1978. In those days memory was scarce and very expensive, so Adam's feat was hailed as nothing short of miraculous. Nevertheless, in my book he stands in Pete Austin's shade.

Pete Austin, Level 9's head Guru, has done wonderful things with text compression - a technique for packing English text into less memory ... one byte.

... former ... Terry Jones. It also features graphics, and is extremely challenging, and up to the best standards of Level 9, who wrote it for Mosaic.

Return to Eden (A, At, B, C, M, S, £9.95) is Level 9's follow-up to their extremely successful Snowball adventure, and the second part of the Silicon Dream trilogy.

Popular

It also includes graphics, and is full of the frustration problems which are Level 9's hallmark ...

... the computer. Be warned, Dungeon Adventure certainly gets ten out of ten for addictive quality!

Level 9 produce games that are usually large ... version of ... adventure ... lot of ...

... become an affinity between ... ure and a good fantasy ... ability to draw the adventurer into a private world. For this reason only adventure succeeds most of Level 9's offerings are a treat added boon they are each available in a good range of the popular micros, most civilised.

Somehow the ... out that ... trum ... versions of ... graphics but ... the 464 as well ... colourful and ... they are too, having ... signed for Level 9 by a professional artist. However, anyone who has played one of Pete Austin's adventures will be well aware, it is the richness of the text and the quality of the imagination behind the plot that makes the ... Eden is a fine ...

Pick of the week

Hard core adventure fans and all persons of taste and discernment will be ...

As an adventure fan, I viewed the prospect of reviewing the latest Level 9 game with excitement. This particular game marks something of a landmark since it's the first Level 9 adventure with graphics and, much to my surprise, the game doesn't suffer adversely. Granted the location descriptions are shorter than before ... assured there ... text.

The game ... Snowball and ... ego Kim Kim ... the Planet E... life. The sc... with about 2... of them illustra... any details of the ... keep you busy for many an hour.

The game is full of tricky puzzles, some horrendously devious. You gain points for solving the adventure, not for collecting treasure.

The illustrations are drawn in multicolour mode and while relatively simple, are well designed and do enhance play. Best of all, they're drawn quickly. The game is accompanied by a very good instruction manual and the usual envelope for a free hint.

Overall, excellent value which, in my view, is unsurpassed by any other software house in this country. Buy it and enjoy. A.W.

CRASH Readers Awards 1984 Winner

BEST TEXT-ONLY ADVENTURE

SNOWB...

by LEVEL 9 COMPUT...

... adventures still represent a large chunk of the ... Level 9 one of the most respected (although ... turned to graphics with their latest release ... this ... hands down, with ...

Problems and puzzles abound; this is a world unknown to humans and many plants or creatures have odd properties. Just trying to survive ten minutes is difficult — Level 9 keep rolling those heavy dice on you but give you a few resurrections before finishing you off.

Exhausted compulsives of the firm's other works may just as well admit to themselves now that they probably won't be sleeping much the next few months. At ... spheric and original.

Eden, another brain scrambling adventure from Level 9 and sequel to Snowball. The presentation has changed — the game includes location graphics, yellow word display on a black background and 'type ahead'. That feature ... input text in ...

The appearance of a new program from Level 9 is a flag-day for all aspiring adventures and, in my household, a signal for the cat to hide under the bed for the duration against the inevitable moment when I go rampaging through the flat, a wild look in my eyes, muttering ferociously about bricklaying birds, nudist beaches and the like. Since *Return to Eden*, the sequel to *Snowball* is out, the cat may be ...

A new title from Level 9 is always reckoned to be an Event, and Pete Austin groupies won't find anything to disappoint them in Return to Eden – which also ...

... for every computer, ... Level 9's other games. In fact, Pete Austin is so prolific in adapting his games to other machines, that I fully expect them to be available on washing machines soon! Briefly, it's fully up to Level 9 standard. That's praise indeed.

Level 9 make no secret of their own 'a code' used ... revise Snowball and boast ... ever-increasing library of ... compression techniques used ... instigate the likes of Return to Eden. These games are ... the length you might expect ... a well-written game and ... addition, user-friendly, ... intelligent and imaginative ... with good descriptions ... ducing a rich atmosphere ...

... one ... by the player ... unique problems ... with the usual Level 9 flair, ... likeness a departure from Level ...

... The game starts with you awoken from cryogenic ... Obviously something is ... you have escaped from ... avoided the now homicidal ... must endeavour to switch back to autopilot, as some aimed the ship at a star. This adventure is devious ... Level 9 are to be congratulated ... writing such an excellent ad... secondly for including a "Help" envelope. If you are ...

	Cassette £9.95	Disk £11.95 BBC/CBM64
COLOSSAL ADVENTURE: The classic mainframe game, with 70 bonus rooms.	☐	☐
ADVENTURE QUEST: An epic puzzle journey through Middle Earth.	☐	☐
DUNGEON ADVENTURE: 40 treasures to find and 100+ puzzles to solve.	☐	☐
SNOWBALL: Immense Science fiction game with 7000+ locations.	☐	☐
RETURN TO EDEN: Vegan SF adventure (Amstrad, CBM 64, MSX and Spectrum versions have 240 pictures).	☐	☐
LORDS OF TIME: Imaginative romp through World History.	☐	☐
ERIK THE VIKING: Game of Terry Jones' book for Amstrad, BBC, Commodore 64 and Spectrum only. All with 180+ pictures.	☐	
	Cassette £6.95	
EMERALD ISLE: Fantasy adventure for all micros. Amstrad, BBC, Commodore 64, MSX and Spectrum versions have 230 pictures.	☐	

I ENCLOSE A CHEQUE/PO FOR £9.95 PER CASSETTE (£6.95 FOR EMERALD ISLE) OR £11.95 PER DISK

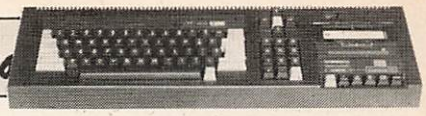
My name: ______________

My address: ______________

My micro is ______________ (one of those listed below with at least 32K, or 64K for MSX graphics games). Send coupon to:

LEVEL 9 COMPUTING
Dept CWA, 229 Hughenden Road
High Wycombe, Bucks. HP13 5PG

AMSTRAD BBC CBM64 SPECTRUM MEMOTECH ENTERPRISE ATARI MSX



Going live on Prestel

MICRONET has launched a major innovation in interactive viewdata – the first live programme on Prestel to be scheduled on a regular weekly basis.

Celebrity Chatline gives micro owners their first chance ever to interview well known personalities direct from their home computers over the Micronet system.

The service is a development of the highly successful Late Night Chatline which is second only to Micronet itself in the Prestel Top Ten of most popular areas accessed.

As Micronet members electronically send questions on special message frames, the night's celebrity replies on-line straight away via his own home computer.

One of the first guests on Celebrity Chatline was Derek Meakin, managing editor of *Computing with the Amstrad* who commented: "It was gratifying for Database Publications to be chosen to help launch this exciting new development in interactive viewdata.

"This is yet another example of the pioneering spirit behind the Micronet operation and helps to explain why micro users are joining in ever-increasing numbers".

Celebrity Chatline is on Micronet 800 every Wednesday between 7 and 8pm.

Amstrad is riding the Stock Market storm

THE nervous mood affecting the computer industry has even rubbed off on Amstrad, star of the 1984 micro market with 200,000 machies sold.

Success of the CPC464 pushed the company's profits up by more than 50 per cent to £9.5 million over the last six months.

Despite this tremendous achievement, stock exchange jitters have meant a drop in share values from 80p to just over 70p at press time.

But Amstrad shouldn't worry about this temporary situation caused by stock market concern over the home computer industry in general.

Survive

The company has the right sort of product to survive "the year of the long hard slog" says industry expert John Rowland, computer merchandising controller for leading high street chain W.H. Smith.

"The trouble is that the city finds it a very difficult market to understand because of the technology involved", he told *Computing with the Amstrad.*

"So as soon as anything adverse is said they tend to throw their hands in the air and rush to get their capital back into areas they know and which will guarantee a 40 per cent return.

"Despite announcing record profits recently, Amstrad's shares have not done well – all because they are now so closely identified with computers.

"But whether the city men appreciate it or not, computers are here to stay.

"This is going to prove a most difficult year for the industry. Not only are the money men in a state of jitters resulting in a loss of confidence, but the market itself is about to be transformed".

Rowland predicts there will be a polarisation of the micro scene which only strong companies with first rate products – like Amstrad – will survive.

"The next stage will see the market divide sharply", he claims. "At one end there will be demand for machines up to around £400 such as the CPC464. Then there will be a huge gap up to the low end entry business machines at around £1,200.

"I see the gap in the middle as the 'black hole' where machines will satisfy neither the home computer fan nor the professional.

Rowland does not see the traumas currently affecting the industry lasting much longer than the end of the year.

"I believe for those who are still around, 1986 will in fact be a very exciting and rewarding year", he claims.

Profits up 50 per cent

In the top four

THE CPC464 outsold all but three other types of computer in the final quarter of 1984 according to a recently released survey.

Amstrad captured eight per cent of the market, behind Spectrum, Commodore 64 and Electron but ahead of the BBC Micro, Commodore 16 and Spectrum Plus.

These seven machines accounted for 86 per cent of the estimated 650,000 micros sold in the final three months of last year.

IN LINE FOR TOP AWARDS

MINI Office, the chart topping business package from Database Software for the Amstrad, has been nominated for The British Microcomputing Awards 1985 in two major categories.

It has been shortlisted for both the Home Software class and Thames Television's Database Home Software of the Year award.

Recognised as the Oscars of the computer industry, The British Micro Computing Awards this year attracted more than 1,000 entries.

Organised by Personal Computer World, The Sunday Times and Thames Television, the awards "seek to define technological excellence and value for money for the consumer".

Mini Office first hit the trade headlines because of its revolutionary price – just £5.95 for a professionally written suite of four programs.

Consisting of a database, word processor, spreadsheet and graphics, it can turn any home computer into an inexpensive office tool.

"We are delighted to have been shortlisted," says Derek Meakin, head of Database. "After all, a truly professional business software package at this price was a gamble – and fortunately it has paid off."

All the shortlisted products are to go before a panel of judges who will then select the top three finalists in each category and ultimately the outright winner.

Chris detects a star prize

IF ever Amstrad's marketing manager Chris Horseman tires of the computer industry he could always use his other great talent and become a private eye.

Horseman proved he can compete with Mike Hammer when he was declared grand winner of the Murder To Go mystery competition at the recent Consumer Electronics Show in Las Vegas.

The contest was organised by software house Infocom to publicise its latest interactive mystery program, Suspect.

Horseman and the other contestants assumed roles as detective, noting their suspicions of whodunnit on a sleuth sheet provided by Infocom.

A theatre troupe masquerading as possible suspects enacted scenes concerning the crime.

After considering a police interrogation of the suspects, an autopsy report and a checklist of personal assumptions the competitors had to identify the murderer, the motive and the weapon used.

Several contestants correctly identified the guilty parties – but only Horseman deduced the means by which the victim was polished off.

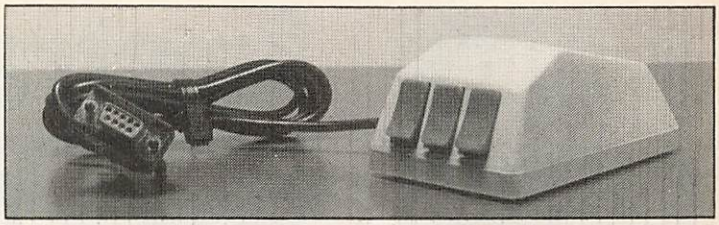
His prize – another Murder To Go mystery competition, this time involving a weekend for two in Bermuda.

"Unfortunately this clashed with my move to a new house so I wasn't able to go", Horseman told *Computing with the Amstrad.*

"However the organisers sent me a nice cheque instead, so it all turned out quite nicely".

Good news for CPC464 owners is that Amstrad is considering making Infocom's interactive fiction software available here.

It runs on CP/M which the Amstrad disc system supports and would be marketed on 3in discs.



The SMC mouse – soon to be available for the Amstrad.

16-colour mouse on the way

A MOUSE complete with icon type graphic software and sprite designer for the CPC464 will be available before June from SMC Supplies.

Phil Sabin of SMC told *Computing with the Amstrad* that the software will be similar to that of the pioneering Macintosh, but offering 16 colours. The Macintosh only has black and white.

Features will include various brush shapes, air brush, fine point, rubber banding, triangles, circles, boxes, fill, variable size text, save/load from tape/disc and screen dump to printer.

The mouse has three buttons, though SMC uses only two with its software, and is connected to the computer via the joystick port. It can be used as a joystick with one or more of the buttons being used as fire keys.

All the software will be in machine code and supplied on cassette with a disc transfer routine. Price is expected to be £59.95.

Interface brings in the robots

AN interface cable which allows IGR's turtle-like robot Zero 2 to be operated by the CPC464 will be on the market this month.

The robot, which sells in kit form for £79.95 and assembled for £99.95, has been produced initially for education and games.

But IGR says over the coming months it will be upgraded to perform simple, useful tasks as well as becoming the basis for a sophisticated robotic games playing system.

Zero 2 is a small robotic device equipped with wheels, pen, lights, line follower and two-tone horn controlled by signals through a cable from the computer.

It is capable of precision movement in forward, backward or turning motions to accuracies of 1mm or one degree, enabling programs to be written to instruct the turtle to draw a trail of complex shapes accurately, says IGR.

Zero 2 can be equipped with a bump sensor so the computer can be informed that the robot has met an obstacle, and can issue commands to avoid it.

Developments are underway to give Zero 2 many more abilities such as speech synthesis and a two way infra-red link that will do away with the need for an umbilical connection.

Zero 2 is primarily intended for three markets:

- Domestic – for education at home and personal recreation and experimentation.
- Schools – for teaching interaction with the computer, logical thinking and as a focus for programming projects.
- Technical colleges – for teaching control theory, robotics, and providing a mechanical base for experimental, electrical design projects and advanced programming.

Spreading the word worldwide

WE are glad to report that *Computing with the Amstrad* is helping to spread the fame of the CPC464 worldwide.

Already the magazine is helping to promote the machine as a teaching aid in Asian schools, and opening up new export markets for UK software suppliers.

Hans E. Kawulusan has written from Indonesia to say how much he has enjoyed the first four issues of *Computing with the Amstrad.*

His institute, Polytechnique Jakarta, is promoting Amstrad in schools in Indonesia and a sister company is negotiating for the rights to distribute the CPC464 throughout that country.

Now he wants all Amstrad hardware and software producers in the UK to contact him at 68 Panglima Polim 1, Jakarta 12160, Indonesia, if they are interested in distributing their products in that country.

Now for the World Cup

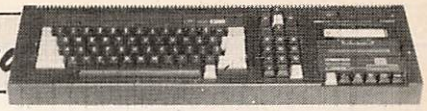
PLAYING for your country in the World Cup can become a regular experience with World Cup Football from Arctic Computing.

Up to nine people can play, each with a different team. Players choose their team from a menu of countries and guide their men round a 3D pitch.

Depending on which team they are up against they play either an opponent or the computer. Price is £7.95.

Jack arrives

ENGLISH Software has released its most popular title Jet-Boot Jack on cassette for the CPC464 priced £8.95, and says it "is the first of many titles" planned for the Amstrad.



FIRST ROM SOFTWARE FOR THE AMSTRAD

ARNOR is claiming to be the first company to produce ROM software for the Amstrad.

The first of several planned offerings is Maxam, a Z80 assembler package which also includes a text editor and debug program.

The 16k eprom plugs straight in but still allows room for the disc unit and has a socket for additional expansion.

It enables the user to include machine code statements in a Basic program automatically assembled when the program runs. Alternatively the text editor can be used to enter assembler statements.

It costs £59.90.

Lightning strikes

ANOTHER major national distributor has joined the Amstrad team.

Lightning has been appointed supplier of the CPC464 to the smaller independent retailers who are currently not serviced by main distributor Europa Electronics.

Lightning's home computer director Loretta Cohen said: "Amstrad's product has already proved itself in the marketplace and new software for it is arriving every day".

Sorcery? It's magic!

THE latest version of Virgin Software's Sorcery arcade adventure aims to cast a spell over Amstrad owners.

Based on an original idea by 16-year-old Martin Wheeler, the Amstrad conversion was done by Dave Chapman who also played a big part in persuading Virgin's managing director Nick Alexander that the Amstrad was worth programming for.

Says Alexander: "I was totally wrong about prospects for the machine. Now I'm having to eat my hat".

Sorcery costs £8.95.

'UNIQUE' HOBBIT FANTASY IS HERE

CLASSIC adventure game The Hobbit is now available for the Amstrad from Melbourne House.

The first computer game to combine complex text with atmospheric graphics, The Hobbit is based on the fantasy land described in Tolkien's famous books.

The player takes the role of Bilbo the Hobbit whose task is to steal the dragon's treasure while faced with many dangers.

Because of its unique features, each time the game is played events proceed in a slightly different way. The player can instruct the Amstrad in ordinary English and can talk to the other characters, answer questions they ask and suggest to them what they should do.

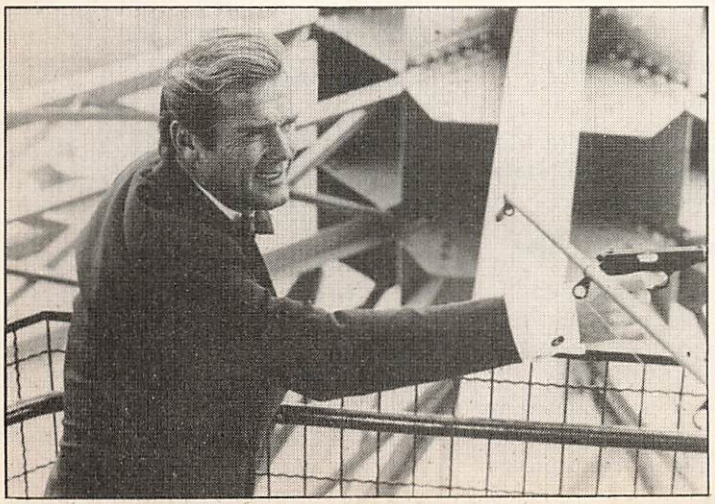
Price is £14.95.

Do your own head checking

THE Azimuth 3000 head alignment program from Interceptor Micros has been developed for the Amstrad.

The kit makes it possible for users to align the azimuth setting on their cassette unit then regularly check and maintain the setting to virtually eliminate loading problems.

Price is £8.99.



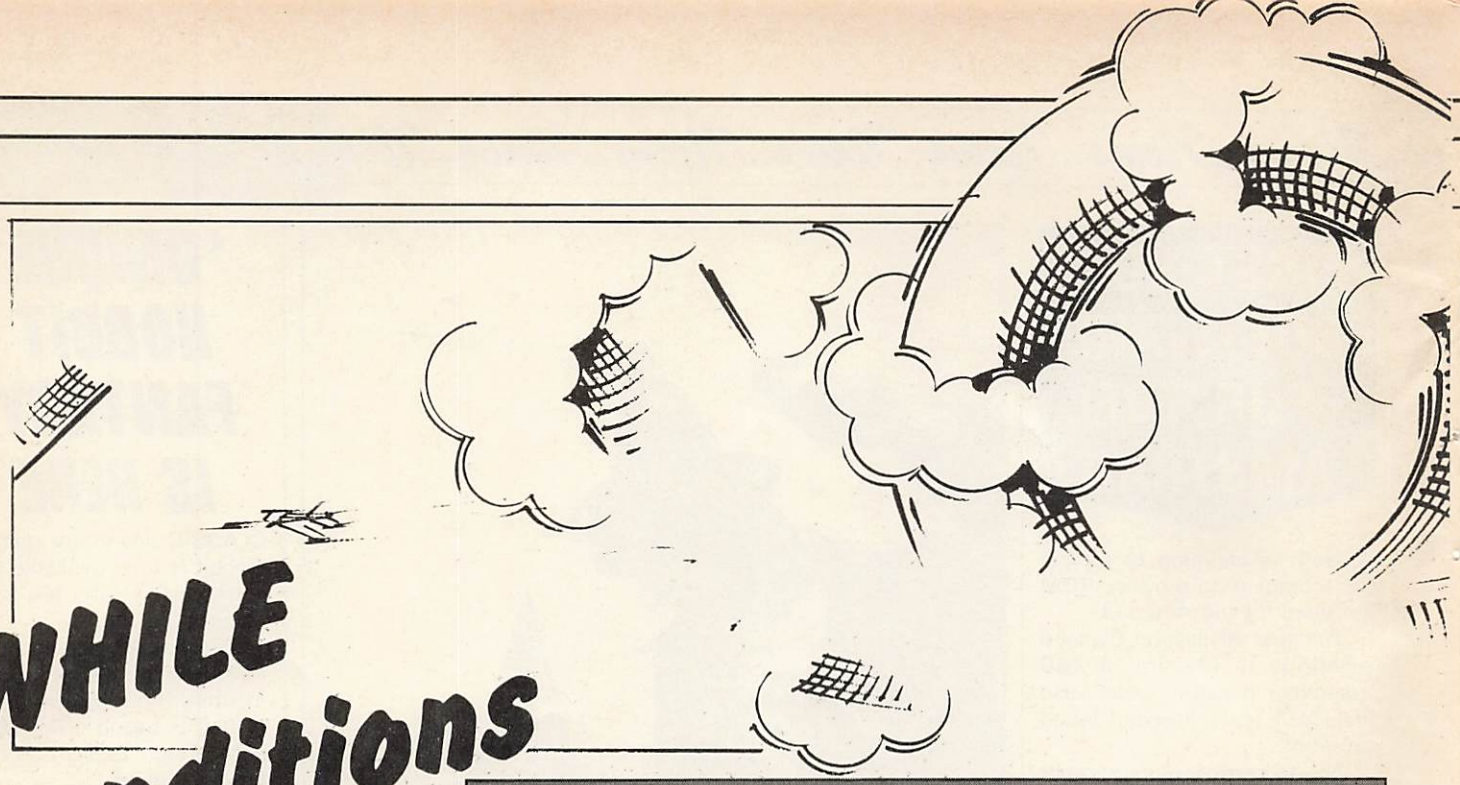
Amstrad picked for 007 debut

MASTER spy James Bond is to be the subject of a computer game for the first time – and the Amstrad has been chosen for his debut.

The latest 007 adventure, "A View to Kill", is being developed for the Amstrad by Domark, which secured the worldwide licence rights against tough opposition from other leading international software houses.

Details of the game are being kept a closely guarded secret but Domark says the program will be centred on four arcade games linked to the major stunts in the film.

Title music from Duran Duran will run through the game, scheduled for release along with the film in June.



WHILE conditions are ripe... WEND your way through loops

Fifth in MIKE BIBBY's helpful guide through the micro programming jungle

THERE is quite an assortment of new ideas this month, which should greatly increase the scope of our programs.

Firstly, if you cast your mind back to last month we saw that the INPUT statement enables the Amstrad to ask for information when a program is running.

When the micro encounters a line such as:

```
10 INPUT name$
```

it halts the program, puts a prompt (?) on the screen and waits for a response.

You then type in what you wish *name$* to be and press Enter. From then on the program continues with the new value of *name$*.

We also saw that it's sensible to print a message on the screen before INPUT to indicate the kind of response required.

Last month we used techniques such as:

```
30 PRINT "How old are you";
40 INPUT age
```

Actually we can incorporate such messages inside the INPUT statement, as in the following:

```
10 INPUT "How old are you"; age
```

Look carefully how it's done:

- You insert the message between INPUT and the variable.
- The message is in quotes.
- There is a semi-colon between the quotes and the final variable.

Using this technique, the program that asked someone's age – Program X last month – would become:

```
10 REM PROGRAM I
20 MODE 1
30 INPUT "How old are you";age
40 PRINT "I don't believe you are ";a
ge
```

Program I

Last month's multiplication program would become:

```
10 REM PROGRAM II
20 MODE 1
30 INPUT "First number";first
40 INPUT "Second number";second
50 PRINT first " multiplied by " seco
nd " gives " first * second
```

Program II

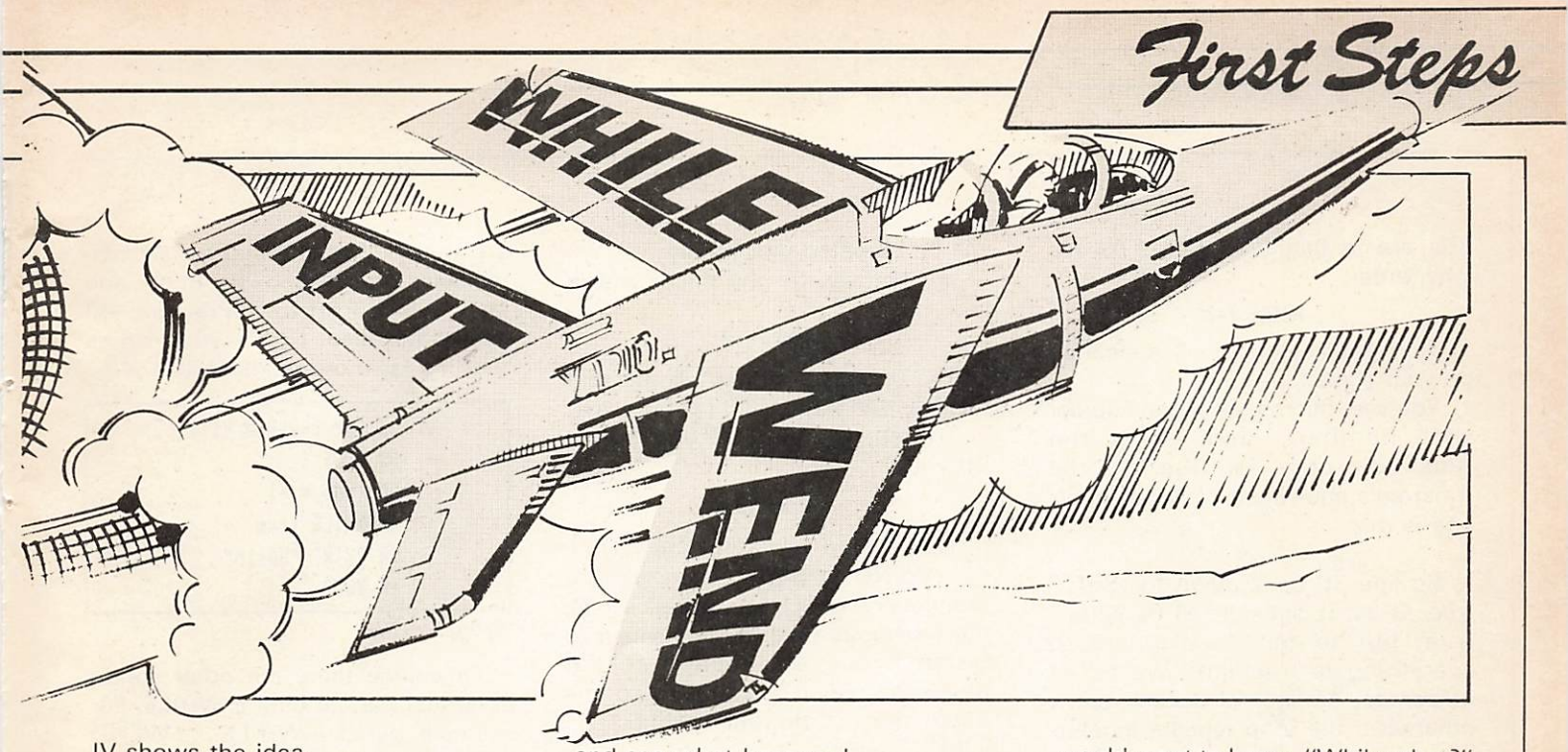
And, of course, this idea of labelling INPUT works with string variables, as well. Last month's Program IX translates as:

```
10 REM PROGRAM III
20 MODE 1
30 INPUT "What is your name"; name$
40 PRINT "Good to see you ";name$
```

Program III

Notice that we don't have to include the question mark in our labels – INPUT automatically supplies one.

Before we leave INPUT there's one other point I'd like you to consider – you can use INPUT to input more than one variable at a time. Program



IV shows the idea.

When you run it, you'll be prompted for two numbers. Type your first number and press Enter.

You'll be met with the message *?Redo from start.* This is because you haven't given the micro enough numbers to fill up the variables it's got

```
10 REM PROGRAM IV
20 MODE 1
30 INPUT "Two numbers"; first, second
40 PRINT first " multiplied by " seco
nd " gives " first * second
```

Program IV

in the INPUT statement.

You have to give the number two values, separated by a comma, just as the variables in the INPUT statement are separated by a comma.

Try doing that now. As you'll see from the output, the first number is stored in variable *first* and the second number in variable *second.*

You don't have to stick at just two variables, provided they're separated by commas, and the technique works just as well for string variables as numerics.

Personally I dislike multiple variable INPUTs – they can lead to a lot of confusion on the user's part. My rule is one INPUT statement, together with appropriate label or message for each variable.

Incidentally the above ideas are why you should type in one thousand as 1000 and not 1,000 as we might write it. As far as an INPUT would be concerned, you're entering two separate numbers when you use the comma. Try entering 1,234 into a line like:

```
10 INPUT number
```

and see what happens!

And now for something completely different. You should remember programs such as:

```
10 REM PROGRAM V
20 MODE 1
30 PRINT "Hello"
40 GOTO 30
```

Program V

This type of program, which endlessly repeats itself, is called an unconditional loop. To get out of it you'll have to press Escape twice. As we learned, such loops are not particularly useful things to have in a program.

Even so, let's have a look at another way of obtaining an endless loop.

Program VI shows the idea:

```
10 REM PROGRAM VI
20 MODE 1
30 WHILE 1=1
40 PRINT "Hello"
50 WEND
```

Program VI

Run it before you try to work out what's going on. As you'll see, Hello is repeatedly printed out.

WHILE and WEND are two Basic keywords that act as "markers" around a section of a program that's to be repeated. We call the lines of program that are to be repeated the body of the loop.

WHILE marks the beginning of the body of the loop and WEND marks the end.

Of course you can't just have WHILE on its own. It would be a bit like saying, "I'll do it while". People would want to know, "While what?".

When we use while in the ordinary sense we always give a time limit or a condition, such as: "I'll do it while you're away".

Similarly we have to give the micro a condition to go with WHILE. Hence line 30 reads:

```
30 WHILE 1=1
```

Here the condition is that we're going to do the loop *while 1=1.*

Of course if I said to you: "I'll do it while one equals one", apart from wondering at my sudden poetic turn of phrase, you'd understand what I was talking about. Since one always equals one, I'd be doing whatever it is forever.

The same is true in Program VI. 1=1 is always true – we say our condition is always met – so we keep on repeating the lines of the program between the WHILE and the WEND. In this case, that means repeatedly printing out Hello.

WEND, by the way, stands for WHILE END, marking as it does the end of the effect of the WHILE.

```
10 REM PROGRAM VII
20 MODE 1
30 WHILE -1
40 PRINT "Hello"
50 WEND
```

Program VII

Program VII is virtually identical to Program VI apart from line 30, which reads:

```
30 WHILE -1
```

In fact the output of the two programs is identical, and the two line ▶

30s are in fact equivalent. To see why, enter:

```
PRINT 1=1
```

The result will be a −1 appearing on your screen.

You see, micros are much happier with numbers, and −1 is the Amstrad's code for true. So the Amstrad's equivalent of *"one equals one is true"* is:

"1 = 1 is −1".

So line 30 boils down to *WHILE true.* Okay, it doesn't tell us what's true, but to our trusting micro everything is true until we tell it otherwise. As line 40 doesn't tell it otherwise, the loop repeats indefinitely.

If the above seems a bit confusing, don't worry too much. Just take it for granted that lines of a program (or code as it's known) surrounded by:

```
WHILE -1
```

and

```
WEND
```

will be repeated indefinitely.

Incidentally, try the micro with a lie:

```
PRINT 1=2
```

The outcome should tell you that the Amstrad's equivalent of false is 0.

Try replacing line 30 of Program VII with

```
30 WHILE 0
```

Now you're asking for the WHILE . . . WEND to be repeated while it (whatever *it* is) is false. As we've said, our ingenious Amstrad believes *it* to be true, and we haven't told it otherwise. So *it* isn't false, so the micro doesn't do the WHILE . . . WEND loop.

In fact since the program is so short nothing happens. If there were a line after the WEND, the program would carry on from there. (We call this dropping through the loop.)

I dislike all this messing with −1 and 0. I much prefer words. So in larger programs where I'm going to be taking into account whether things are true or false, I tend to use two variables, *true,* which I set to −1, and *false,* which I set to 0. Program VIII shows the idea. I think you'll agree that it's far more readable this way.

```
10 REM PROGRAM VIII
20 MODE 1
30 true = -1
40 WHILE true
50 PRINT "Hello"
60 WEND
```

Program VIII

Of course there are other conditions that can go with a WHILE. For example, here's a WHILE . . . WEND loop to read a book:

```
WHILE still another page
read next page
WEND
```

Here our loop continues on the condition that there is still another page.

Program IX uses a WHILE . . .

WEND loop with a condition. Lines 40 and 50 are to be repeated while *finish$* is not equal to YES. That's what the symbols < > mean – not equal to.

Line 30 would translate *WHILE finish$ is not equal to YES.*

Each time you come to line 50 you're asked to INPUT a string into

```
10 REM PROGRAM IX
20 MODE 1
30 WHILE finish$ <> "YES"
40 PRINT "This is a WHILE...WEND loop
"
50 INPUT "Do you want to finish"; fin
ish$
60 WEND
70 PRINT "Then I quit!"
```

Program IX

finish$*. If you enter anything other than YES the loop will keep on repeating, since the condition for doing the loop (given in line 30) is met – *finish$ is not YES.

As soon as you respond with a YES to the prompt you've "violated" the loop's terms, and the micro stops repeating the loop. It drops through the loop and continues with line 70.

We'll see lots more of WHILE . . . WEND loops with conditions as this series progresses, but for the moment let's turn our attention to Program X.

```
10 REM PROGRAM X
20 MODE 1
30 number = 0
40 PRINT "number is "; number
50 newnumber = number + 1
60 PRINT "new number is ";newnumber
```

Program X

What happens is that we set the numeric variable *number* to zero, then print out its value (lines 30,40).

We then add one to *number* and store it in another variable, *newnumber* (line 50) and then print that variable's value out (line 60).

Let's examine line 50 in detail:

```
50 newnumber = number + 1
```

When the micro reaches this line it takes the value stored in *number*, adds one to it, and stores the result under the label *newnumber*.

Notice that the micro does the sum that appears on the right of the = , then stores the result in the variable on the left.

```
10 REM PROGRAM XI
20 MODE 1
30 number = 0
40 PRINT "number is "; number
50 number = number + 1
60 PRINT "number is now ";number
```

Program XI

Program XI is based on Program X. In fact it does much the same job, printing out zero, increasing it by one and then printing out the answer. This time, however, instead of using two variables, we make do with one, *number*.

Can you see what's happening in line 50? Remember, we do what's on the right of the = , then store the result in the variable on the left. So when the micro encounters:

```
50 number = number + 1
```

it takes the value labelled by *number*, adds one to that value, then stores the result back in the variable *number*. In other words, line 50 increases the value of *number* by one. Had line 50 read:

```
50 number = number + 2
```

The value stored in *number* would have been incremented by two. Try it!

Let's get something clear: mathematically:

```
number = number + 1
```

does not make sense. How can a number be equal to itself plus one? It's like writing:

```
5 = 5 + 1
```

However we're not using the = as an equals sign. We are using it to signify assignment – that is, we're using it to pin a label, or variable, onto something. When we write:

```
newnumber = number + 1
```

we are telling the computer to do the job on the right – that is, to add one to the value labelled by *number* – then to label the result of that sum with the label on the left of =.

The only reason that:

```
number = number + 1
```

might seem confusing is that we are labelling the result of the calculation on the right with a label that's already been mentioned in that calculation.

That doesn't worry the micro though. It's used to re-using the same label. That's why they're called variables – they keep varying!

```
10 REM PROGRAM XII
20 MODE 1
30 number = 0
40 WHILE -1
50 number = number + 1
60 PRINT "number is now ";number
70 WEND
```

Program XII

Have a look at Program XII. Line 50 is identical to that of Program XI. That is, it increases the value of the variable *number* by one.

However line 50 is part of a WHILE . . . WEND loop. Hence, no sooner has it incremented the number (line 50) than it is printed out again (line 60), incremented again (line 50), printed (line 60) and so on.

The outcome is that line 60 prints out a steadily increasing sequence of numbers.

Try altering line 50 of program XII so that the numbers go up in 2s, 4s and 10s. Could you, by altering lines 30 and 50, get the program to start at 1000 and count down in ones?

We can actually get the program to stop at a given number by altering the condition in the WHILE statement. Try altering line 40 of Program XII to :

```
40 WHILE number <> 24
```

This should give you the numbers 1 to 24 on the screen. Leaving this altered line 40 in our micro, swap lines 50 and 60. That is, make line 50 into line 60, and line 60 into line 50. The effect is that now we increment *number* after we've printed it out.

Now run the program. The output should be noticeably different. Can you explain why?

I should also point out that we don't strictly need line 30 to make *number* zero. The Amstrad assumes numeric variables are zero until told otherwise.

However it's good programming practice to put this line in – a little superfluity can go a long way towards improving program clarity.

That's all for this month. Next month we'll have a look at some more loops.

POOLSWINNER

THE ULTIMATE POOLS PREDICTION PROGRAM

Boxed, with detailed instruction booklet

- **MASSIVE DATABASE** Poolswinner is a sophisticated Pools prediction aid. It comes complete with the largest database available – 22000 matches over 10 years. The database updates automatically as results come in.
- **PREDICTS** Not just SCOREDRAWS, but AWAYS, HOMES and NO SCORES.
- **SUCCESSFUL** SELEC guarantee that Poolswinner performs significantly better than chance.
- **ADAPTABLE** Probabilities are given on every fixture – choose as many selections as you need for your bet. The precise prediction formula can be set by the user – you can develop and test your own unique method.
- **SIMPLE DATA ENTRY** All English and Scottish team names are in the program. Simply type in the reference numbers from the screen. Or use FIXGEN to produce fixture list automatically (see below).
- **DISC/MICRODRIVE COMPATIBLE** All versions (except Apple and IBM) are supplied on tape, with simple instructions for conversion to disc/microdrive operation.

(This seasons results are supplied with the package so that predictions can start immediately.)

AVAILABLE FOR Spectrum (48K), Commodore 64, VIC 20 (+16K), AMSTRAD CPC 464, BBC B, Atari (48K), ZX81 (16K), Dragon, Apple II, IBM pc, ELECTRON

PRICE £15.00 (all inclusive)

NEW

FIXGEN 84/5

AT LAST: No more struggling for hours to get the fixture list into the computer. FIXGEN has been programmed with all English and Scottish fixtures for 1984/5. Simply type in the date, and the full fixture list is generated in seconds. Fully compatible with Poolswinner.

POOLSWINNER with FIXGEN £16.50 (all inclusive)

Fixgen alone £5.50 (yearly updates available)

COURSEWINNER V3

THE PUNTERS COMPUTER PROGRAM

Coursewinner is designed to allow you to develop and test your own unique winning system. Using information from daily newspapers or 'Sporting Life', the most important factors can be input and analysed. The program is supplied with a database detailing best trainers and jockeys, and effect of the draw for all British courses. (Flat & National Hunt.)

AVAILABLE FOR Spectrum (48K), Commodore 64, BBC (B), AMSTRAD CPC 464, Atari (48K), Apple II

PRICE £15.00 (all inclusive)

AVAILABLE (RETURN OF POST) FROM . . .

VISA phone 24 hrs

selec SOFTWARE

Access phone 24 hrs

37 COUNCILLOR LANE, CHEADLE, CHESHIRE. ☎ 061-428 7425

OVER 130 AMSTRAD CASSETTE TITLES OVER 90 NOW TRANSFERRED TO DISC ALL NOW IN STOCK

SOFTWARE for CP/M–
Macro 80, Microsoft Basic, Microsoft Basic Compiler, other titles on request.

KUMA full range of latest titles including, Music Composer, Zen Database.

Full Business Software range for **£39**

● TAPE TO DISC TRANSFERS ●

Mail order welcome, P&P free of charge
Please send sae for full list to:

TIMATIC SYSTEMS LTD

Registered Office:
NEWGATE LANE
FAREHAM, HANTS PO14 1AN
Tel: FAREHAM (0329) 239953

Sales and Repairs:
FAREHAM MARKET
FAREHAM, HANTS
Tel: FAREHAM (0329) 236727

Announcing MAXAM for the AMSTRAD CPC464

SOFTWARE in ROM!

The start of a complete Expansion System . . .

SIDESWAYS ROMS at last!
No more loading . . .
Leaves 40K free!

The perfect system:
* All-powerful Assembler
* Complete Disassembler
* Full screen editor
* Multi-function Adaptor
* Huge expansion potential in one simple unit!

So easy to use and learn . . .

```
10 MEMORY HIMEM-10
20 start=HIMEM+1
30 !ASSEMBLE,start
40 'get start
50 'limit &FFFF
60 'ORG start
70 'CP 10:SCF:RET Z
80 'RST 1,&B752
90 'ORG &BD2B
100'JP start
110'END
```

Meet MAXAM – a new full-feature no-compromise Assembler/Disassembler/Editor – with a difference. It's in a very full 16K EPROM which plugs directly into the AMSTRAD. No waiting while it loads – it's always there! You can still use the Disc unit. You also get, as a bonus, a new expansion socket for Arnor's new range of Sideways ROM cartridges (containing, for example, our forthcoming Word Processor).

MAXAM uses no BASIC RAM space. It lets you mix BASIC and Machine Code – just like the BEEB! Or, you can assemble direct from the Editor, and you can even use the Editor to edit BASIC programs!

MAXAM is ESSENTIAL software for the AMSTRAD enthusiast.

Cassette (reduced specification): £13.50
Disc: £26.90. All prices include p&p.

MAXAM in ROM £59.90

Arnor LTD
High Quality Software

Software Houses: We have the perfect low-cost system for software in ROM! Talk to us!

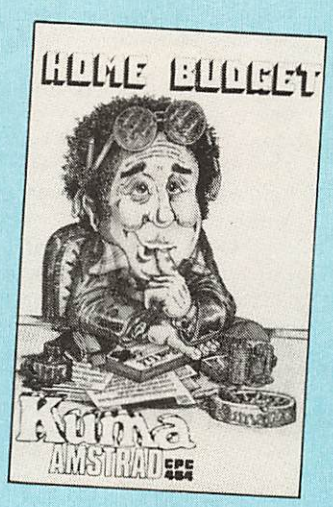
Technical Data

*Super-fast 3000 lines/min assembly *Conditional Assembly *Plain English error messages *Full Expression evaluation *Unrestricted label names *Directives include: ORG, BYTE, WORD, TEXT, RMEM, LET, IF, GET, PUT, LIMIT, CODE, NOCODE, READ. Commands include: LIST, NOLIST, LISTP, TITLE, PAGE, PLEN, WIDTH, DUMP. *Menu-driven Screen Editor includes move copy and delete block, tabs, search and replace, print all/part of text, Load/Save all/part of text. Disc/ROM version only: Register display, Memory Edit commands, breakpoint, string search in RAM. Link to AMSDOS.

Technical Enqu. 01-852 2174

Cheques/P.O.s to: **Arnor Ltd, PO Box 619, London SE25 6JL. Order Hotline 01-653 1483 (2pm-6pm)**

Easy way to keep your home finances in good order



HOME Budget from Kuma is designed to enable you to create and maintain annual files containing estimated monthly expenditure and income.

The program is driven by two menus. The first gives you a choice between creating an initial file, or loading an existing file. Whichever option you choose you are presented with the second menu which has eight options.

Selecting the first will display 12 categories of expenditure. These cover mortgage, rates, insurance, rental, heat/light, housekeeping, clothes, entertainment and others, which can be used or amended to suit individual requirements.

This option will later display six categories of income. These are initially salary, expenses, fees, sales, tax rebate and sundries – though the headings can also be altered to suit the individual.

The new file must be opened by entering the month and year of the starting date. The ending month should be 11 months later, so for a file starting in June you enter 6 to start and 5 (May of the next year) as the ending month.

A positive or negative number can be entered into the opening month as a carried forward figure.

Enter the forecast figures as appropriate by using the on-screen options for moving forwards and backwards through the year. Once this is complete, a similar operation can be carried out on the income items if required.

When all the information has been entered you are returned to the main menu from which the various display options can be selected to verify the contents. The information can be edited where necessary before finally saving it to tape.

A flashing screen prompt on the main menu indicates that a file created or amended has not been saved. When saving you may record, if you wish, the date and time but you *must* input a six figure security code and a file name. For obvious reasons these must be remembered when loading the file at a later date.

Having saved the file you must go through a preset verify procedure. If a read error is encountered the data can be resaved by returning immediately to the main program where the information will be found intact – a very useful touch.

The files can be updated at monthly intervals using the "write to a file" option and entering information from your bank statement, cheque book stubs and so on – turning forecast into fact.

After this option the program will display the new figures together with the forecasts which can then be changed if required. Any surplus between the forecast figures and the actual figures is held in a buffer for amending the following month.

At six monthly intervals, or longer if desired, the file should be loaded and the "open subsequent file" option called. After all, much of the present file will be past history, and you'll want your forecasts to cover more of the future months.

A new month number should be entered which must be ahead of the month on the first file.

The screen displays are superb, neatly laid out, and colour is used to great effect. All entries are well error-trapped and the menus enable you to toggle backwards and forwards between screens of information. The whole package is extremely professional and easy to use in its presentation.

My only criticism – a minor one – concerns the instruction card. The writer appears to have missed out the option to load a file and as a result each of the remaining options is one number out. This is a small error which I am sure will soon be corrected, and anyway the screen prompting is so good that the card is redundant after a first cursory glance.

Alan McLachlan

... and losing money without tears

I'VE always wanted to gamble recklessly and not be in the slightest danger of losing money, so when the opportunity came to review **Fruit Machine** from Amsoft, I jumped at it.

I have played several other versions on different machines and this one came out of the comparison reasonably unscathed although it did have its weak points.

The screen format is what you would expect with the machine's main window taking up about half the available room. The remainder of the screen is used for passing information about your various options.

The reels are set in motion by a press of the space bar and what immediately follows is the disappointing part.

Instead of the winning symbols scrolling through each window, they are replaced by a roller shutter blind of coloured stripes. These just flash different colours until the time has come for the winning line to be displayed. A variety of the usual bells, grapes, plums and lemons – sorry fans, no cherries – are displayed on screen once the time is reached to show the winning line. These are interspersed with some £ signs, stars, lucky 7s and bars which all have their own values.

As with the mechanical version, certain combinations pay more than others and the various winning combinations are listed on the cassette inlay.

Surely it wouldn't have taken much more effort, time and space, to have included ▶

these in a colourful on-screen display, perhaps while the main program was loading.

You start the game with £5 and every spin of the machine costs you 10p. The display shows the amount the machine has paid out in winnings and the amount you have inserted as two separate totals.

Your game ends automatically after you have spent your fiver whether you are winning or not – your winnings consequently counting as your score for the game.

A winning line presents you with the alternative of either a money prize or a number of nudges.

There are several options available before or after each spin. The first is to hold certain reels, which is done by pressing the number key relating to the reel you wish to hold.

The second is the ability to gamble a win – money or nudges – in the hope of gaining the next higher prize.

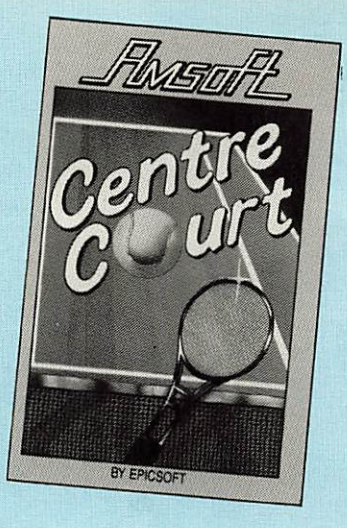
The final special is the Winner Spinner. If you get one – and you can get several depending on the size of your normal cash win – you can opt for a spin instead of cash.

Should you risk this you get another winning line along with its prize. This can sometimes be more, or sometimes less than your original win. These Winner Spinners can also be gambled if you so desire.

Although more could have been made of some of the ideas in this game, the same I'm pleased to say cannot be said for the sound effects. These are excellent.

This particular version did not really appeal to me, as I like to gamble everything I've got including my winnings. Even so it is entertaining and should appeal to all ages, of whatever financial standing.

Paul Murphy



Faster than you think on court

LOAD **Centre Court,** by Epicsoft, and you have three options – play the computer, another player or "watch Roland v Arnold in an enthralling match, demonstrating the capabilities of this game".

Seems sensible to have a look see, and you do. After five minutes you decide you've got the gist and look around for some way to get back to the main menu.

Ten minutes later you decide there isn't any way back apart from waiting for the end of the demo. And life's really too short for that.

So reload, and try playing the computer. Up comes the court again, complete with grass, scoreboard, spectators' heads – yes, turning to follow the ball, linesmen and umpire. Quite neat.

You can use joysticks or keys – numeric keyboard for one player, A, D, Z, C, W for the other in the two player option – and after sampling both I found the joystick came easier to hand.

I was quite surprised to hit the ball first time. The computer served first, and helped no end by making a few mistakes itself.

As the ball flies off you can see its shadow on the ground, an indispensible aid in judging the ball's height from the ground.

When a player wins a point the crowd cheers lustily, if a trifle harshly. They spent most of their time cheering my bloomers.

Having played a couple of hours I found serving the hardest part. I'm glad the program didn't make provision for top spin, slice and the rest of the stars' repertoire.

When you first see the game you think everything is moving far too slowly. Then you try it and find it is you who are lagging behind. Any faster and it would be impossible for me to play.

I'm sure I would have had an even better game if someone had accepted my challenge. But even MacEnroe chickened out.

Jed Glover

Go for fame

THERE can't be many people who haven't dreamt of being rich and famous. Now there's a chance to escape onto the next screen, but this cannot immediately be achieved.

There is a time delay, dependent on the level selected, before the exit route appears, and this time can be used in frantic evasion from the marauding pursuers or in gaining extra points and revenge with one's laser.

Screen 1 is easy to survive, being set in the front room, with dust being the moving enemy. Passing on to the bathroom, the showerheads produce dangerous floating bubbles which occasionally fragment and become even more tricky.

The garden contains spiders with poisonous strands, although their movements are fairly predictable.

The guard room has bright red robots which emit bouncing rays. This screen is the first to feature a cavern, which requires a little skill to manoeuvre.

Finally we move on to the Dragon's Lair, with a magnificently designed multi-coloured dragon guarding a pile of gold.

The successful player here will grab some gold and escape through the doorway and back to the first screen.

The program contains some nice graphics, smooth action and fair use of the Amstrad's sound facilities. It isn't particularly original, but it does seem to have that kind of addictive quality that becomes so annoying.

Phil Tayler

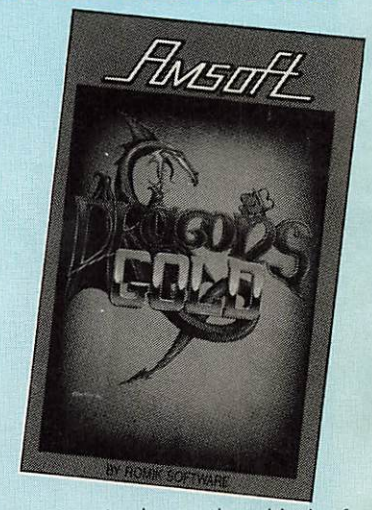
Minus originality, plus addiction

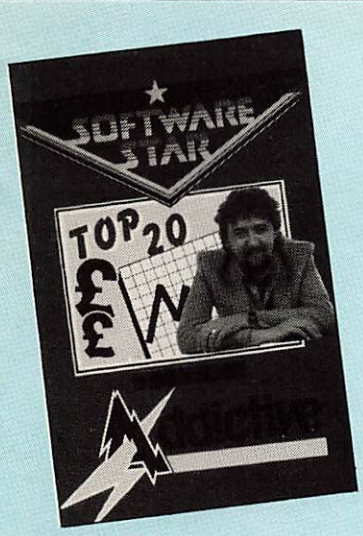
ONE of the current ideas dominating software writers seems to be to think of the most unlikely objects to be cast as the villains. Perhaps readers might have been pursued already across computer screens by anything from telephones to toilets.

Dragon's Gold, however, written by Roland Stokes (surely not the Roland?) and produced by Romik Software seems to have little to commend it as far as originality is concerned.

There are six screens, each holding some static objects of danger, along with some well-programmed sprites.

Each screen shares the same basic purpose, which is





chance in **Software Star** to try your hand in the cutthroat world of computer software marketing without any of the risks associated with running your own company.

In this latest release from Addictive you have been employed as both chief games designer and company chairman and must try to become a Software Star.

This is achieved by getting your games into the top three best sellers. A number one hit will give you the biggest boost to your rating and if you don't make it to this level for a few months the public will soon forget your name.

You are given an initial target of 10 months of fame. As I soon found out this is not easy to achieve. Not only to do you have to make it to the top, you also have to meet the profit targets set by your directors.

At the start of every month you have the option of launching a new game and if you decide to create a new masterpiece the computer will ask you for its name and will tell you what the press reviews had to say about it.

At my first attempt I let my imagination run wild and typed in the name of my game, only to have it rejected because it was too long.

Well I suppose Megagalactic Llamas at the Edge of Hyperspace was a bit over the top!

After the game's launch you have to make decisions about your employees, marketing strategy and sales drives.

Maybe I'm too soft, because offering incentives to my staff actually reduced their productivity and being honest in my advertising lowered the company's reputation.

At month's end the charts are shown and the computer displays the progress of your games.

After a few months a game will have exceeded its saleable life and keeping it on your books will cost you several thousand pounds a month so you would be well advised to drop it.

The optimum level of advertising is higher in the winter than in the summer.

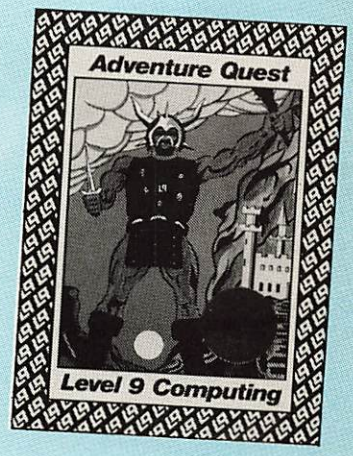
If you book too much advertising, it will cost you dear without increasing your sales beyond a certain point. Getting the balance right is difficult, and despite having a best seller for several months the bank called in the receivers and closed my company.

The game doesn't make great use of the excellent graphics or sound facilities of the Amstrad, nor does it offer the facility of a two player game.

Despite these criticisms the program is fun to play and makes a pleasant change from arcade games. One thing's for sure, I'll never make it as a top games designer.

Steve Lucas

Adventure at its very best



ADVENTURE Quest is the second in Level 9's Middle Earth trilogy and is yet another excellent adventure.

The demon Lord Agaliarept (there must be an anagram in there somewhere) is ravaging the land. In desperation the King instructs the Wizards' Guild to use their powers to defeat him.

You, an apprentice magician, are summoned before the Wizards' High Council. The High Wizard tells you of their plans to mount a magical assault on the Dark Tower, home of Agaliarept.

He has also, however, decided to carry out an undercover operation. You are to try to infiltrate the Dark Tower and kill the evil demon.

Despite your protestations, you are hustled away to the nearest tele-portal and materialise on an all too familiar North/South road.

You begin your quest by entering a nearby building to find several useful items. Don't dismiss any of them, you'll need them all.

A careful search of the locale will doubtless have you lost in the forest. Yes, it is a maze! And yes, you can, in fact better had, map it. You might have a bit of difficulty with the wolves however.

The orchid should not prove to be too much of a problem and you will find you have more than one leg to stand on.

If you can please the unicorn you will detect the sweet smell of success.

Then it is off to the desert and your first major problem – a sandworm. I doubt if you'll get much of a look at it though.

A quick bit of mapping should find you, hotly pursued, running past the pyramid. You know what they say about setting a . . . !

The snakes all came out of a basket and once you have got into the rhythm you will find the temple priestess is glad to see you.

After a quick look around you'll soon be wondering what to do with the oil – gotta lotta bottle – but not yet!

The djinn is just a big bag of wind so ignore him for the moment. Ali Baba will crop up soon, and then it is back to the temple by way of the well for your reward.

Then we go up the bend and Goliath has definitely learnt his lesson. You'll soon meet another friend – he doesn't like orcs since they put him on the ▶

REVIEWED SO FAR

Adventure Quest	Level 9	Map Rally	B.E.S.
American Football	APS	Manic Miner	Software Projects
Amstradraw	B.S. McAlley	Message from Andromeda	Interceptor
Astro Attack	Amsoft	Mr Wong's Loopy Laundry	Artic Computing
Amstrad Tutorial	Amsoft	Mutant Monty	Artic Computing
Atomsmasher	Romik	Number Painter	ASK
Blagger	Alligata	Osprey	B.E.S.
Centre Court	Epicsoft	Punchy	Mr Micro
Chopper Squad	Interceptor	Pyjamarama	Microgen
Colossal Adventure	Level 9	Roland in the Caves	Amsoft
Country Cottages	Sterling	Roland Goes Square Bashing	Durell
Crystal Theft	Wiccasoft	Rollaball	Timeslip
Cubit	Mr Micro	Royal Quest	Timeslip
Detective	Argus Press	Screen Designer	DJL
Dragon's Gold	Romik	Snooker	CDS
'Er*bert	Microbyte	Snooker	Gem
Flight Path 737	Anirog	Snowball	Level 9
Football Manager	Addictive	Software Star	Addictive
Fruit Machine	Amsoft	Survivor	Anirog
Fruity Frank	Kuma	Star Avenger	Kuma
Forest at World's End	Interceptor	Test Match	CRL
Galaxia	Kuma	The Moors Challenge	Timeslip
Gems of Stradus	Kuma	Timeman One	B.E.S.
Ghouls	Micro Power	Timeman Two	B.E.S.
Happy Letters	B.E.S.	Trial of Arnold Blackwood	Nemesis
Harrier Attack	Durrell	Wordhand	B.E.S.
Happy Writing	B.E.S.	Wordwise	B.E.S.
Holdfast	Kuma	Xanagrams	Postern
Home Budget	Kuma	Z80 Assembler, Disassembler and Editor	Arnor
Hunchback	Ocean		

chain-gang.

The orcs get stoned and you will need to throw yourself into the next problem. Now you can get Aladdin's lamp, so on to the Snowman's abode for the next stage of your quest.

Let's hope that if you get dropped in it you can drop back out of it. I'll let you figure out how to get two lungs – or is it get lungs twice?

As you can see from this small part of the adventure I have described, this is one of phenomenal proportions.

It is full of witticisms, bad puns, devilishly twisting puzzles and saturated with atmosphere.

A brilliant adventure that confirms Level 9's place as the ultimate adventure software house.

Paul Gardner

Pictures from stock

DJL's **Screen Designer** consists of a cassette and 21 pages of notes explaining the many functions available. This excellent utility will allow you to create impressive pictures which can be stored on disc or tape.

The program is menu driven, allowing you to edit a picture, change mode, redefine the colours, save or load a screen and save or load graphics characters.

Taking the edit option first there are two modes of operation – pixel mode and text mode. In pixel mode there are 16 different functions. As you would expect there are the usual facilities for plotting points, drawing lines and circles, filling areas, changing pens, paper and border colours.

There are also some advanced features not usually found, such as the ability to magnify any portion of the screen by either a factor of 4 or 16 for highly detailed work on small sections. When work is completed the section can be reduced to normal size and the rest of the screen restored.

The screen can be pixel scrolled in any direction enabling an object to be drawn first and positioned later.

Two lines are taken up by an information window. This is used to show the pen, paper and function selected.

Moving into this area with the cursor causes it to move out of the way. If it was at the top of the screen it moves to the bottom and vice-versa, allowing you to draw underneath it. Delete makes it disappear altogether so you can view your masterpiece in all its glory.

Selecting text mode gives you another 15 functions enabling you to print in normal and transparent mode at any pixel.

All the keyboard characters are available plus a few of the graphics characters, and any section of the screen can be defined as a multi-coloured graphic character and printed at any position.

Going back to the menu (which incidentally doesn't lose you your picture – there is a copy in the memory somewhere) the second option allows you to select mode 0, 1 or 2, so you can select the resolution you require and the number of colours available.

The third option enables you to redefine any of the colours. The number depends upon the mode and these are selected from the full 27 possible on the CPC 464.

The last two options allow you to load and save either the whole screen, or just the graphics characters.

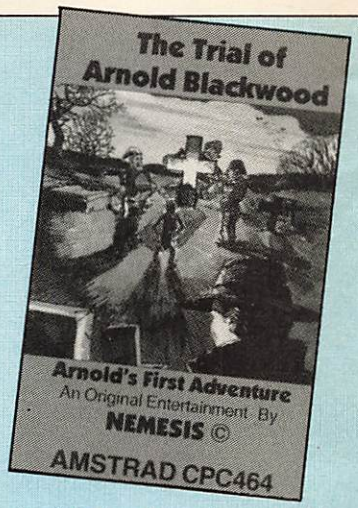
The screen and graphics can be loaded back whether the screen designer is there or not, so it is possible to design the graphics or the screen for your own program and load it to any spare section of memory.

To restore a picture either select edit mode if you are using the designer, or call the address it was loaded to if it is not present.

DJL's Screen Designer is a superb piece of software with excellent documentation and every function you could possibly wish for. I can highly recommend it.

Roland Waddilove

Wobbles with wellies



LORD Erebus has hired you in **The Trial of Arnold Blackwood** (Independent Software) to recover his gold amulet and also rid the grounds of his manor of dangerous flora and fauna.

Almost all commands are just simple three letter VERB NOUN so, having found a flashlight, I spent over an hour trying to get some light in the house.

My mistake was thinking Arnold wore only a raincoat and wellies so FLASH ON would probably cause Lady Erebus to swoon or throw something.

Having seen the light I then had trouble getting rid of the corpse of the rabid dog – surely the best thing would be to cremate it along with the triffid?

Having finally laid Bonzo to rest, put the coal in the bunker and dumped almost everything else on cue at the feet of Lord Erebus I was let out with a warning that there are two more adventures to solve.

The game has about 100 rooms, but 50 are empty, and the Save could dry paint.

At times Arnold says he is "thinking" or "sorting data" and responses are slow – it must be in Basic.

The most irritating thing is that whenever you want to DROP an object (and therefore first take an Inventory) you always have to press (any) key to restart – I was continually trying to ROP things.

The game would totally baffle an American with its peculiar sense of humour. Is the Venusian piano being played backwards?

Albert Fibby

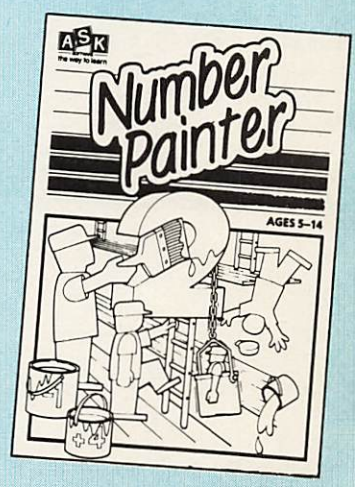
Get up early to be a painter

ONCE I'd loaded **Number Painter** from Ask Software, I let my nine-year-old daughter get at it before me. She was still there at the keyboard some two hours later at bedtime, asking for just one more try.

Then my eldest daughter, aged 17, and her boy friend took over and my chances were gone until the next day. Even then I had to get in quick before my nine-year-old got there before me (actually I'd hidden it).

The program is educational and in two parts, the first of which is an arcade game.

The second is a self test option in mental arithmetic against the clock.



The main program is a levels and ladders game aimed at the five to 14 age bracket. The hero is a painter, beautifully animated, who has to paint numbers to match a

displayed target figure.

This target figure is selectable at the start of the game as a difficulty level.

Numbers from 1 to 9 are displayed on screen preceded by a +,—,×, or ÷ sign (the × and ÷ are highlighted in a different colour).

The Space bar must be pressed when the painter is over the particular number that you wish to paint and take into account towards the total.

Our hero has four incarnations called Mr Plod, Mr Walker, Mr Swift, and Mr Speedy and selecting one sets another level of difficulty in the speed at which he moves around the screen. It is an extreme test to achieve success on the higher levels.

Each level is time controlled as a bucket gradually rises from the bottom of the screen. Once at the top it spills paint all over the screen and our hero has failed.

There is also the occasional gap in a platform, and falling through one of these results in a temporary stunning of our friend as time slowly runs out.

Success with two or three painters will result in a change of scenery as you go up a level, but success with one only will result in his remaining on the same screen.

The graphics are quite exceptional with smooth animation, and good use is made of sound. They keys are easy to use and the game supports joysticks.

The multi-level self-test option is also entertaining and is designed as either a monitor of your progress after playing the game, or a practice mode before playing it.

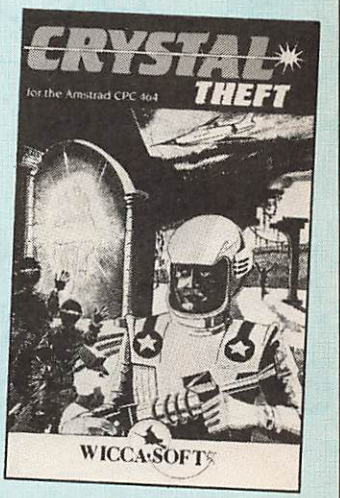
The same little animated character nods or shakes his head according to your answer to a series of problems, which may range from a simple sum on level 1 to a more complex one on level 12.

Both programs uphold the Ask tradition of high quality educational software. If you buy the package your child will certainly gain a better understanding of numbers.

But if you want to have a go on it yourself you had better be prepared for a long wait, or get up early in the morning.

Alison Hughes

Where the cookie crumbled



ANTICIPATION is sometimes the better part of adventuring and nowadays I read instructions before I play. A cassette inlay plus a booklet explained all about the Vegan War and that I was to play a simulation of one of "our side's" major successes.

Slightly disappointed that I was not doing the real thing, but nevertheless all agog, I plunged in.

"I have landed", said the screen. Not only a reconstruction, but it's not even me doing it!

Crystal Theft from Wicca-Soft is a "Find the temple and steal the crystal" text-only adventure. Non-colour owners are not missing much as the display is green.

But it is nicely laid out, with four windows giving location description, inventory, messages from him and suggestions from you.

You start armed with a sonic pistol and — hands up anyone who knows what a katana is?

I usually just amble around on my first go, getting the feel. Idly trying the Store command, I found I'd earned 110 points — heaven knows how.

A Time command tells you how long you've been at it and, although there's no time limit to the total game some problems do have a time factor.

You move N, S, E or W, with the occasional Up and Down, and there's a Save and Restore at certain points. Examine is given for free and so far has revealed a mountain of useless information. But there's a few nuggets in them thar hills.

Promises to understand commands like "Go South then take the key and climb the rope" à la Hobbit seem to be rashly made.

Descriptions are nicely atmospheric and I looked forward to a more serious attempt. And that's where the cookie crumbled. Exits are not shown so movement is by trial and "you can't go thataway".

Unhelpfully, your command is erased as the answer appears. Provoke a "Pardon" and it won't even understand Score or N for several goes.

If the location says "standing outside a door", you should not be told there isn't one there when you say Examine Door.

And I might just accept that West or South takes you to the same place, but West taking you four locations North stretches my goodwill. The author's either a genius or doesn't know his ay from his ee.

Some problems rely too much upon luck, and pre-testing by someone who did not know the solutions would have revealed the pitfalls and the awful Pardon bug.

But even so I enjoyed playing and think, with a little more effort, this author has a future.

Oh, that katana is a Japanese sword — Quincy's body was killed by one.

Dorene Cox

INVASION IN SLOW MOTION

NOW here's an original story line. Earth has been invaded by aliens and you, Captain Macho, are the only person who can save humanity.

By assembling nine fighter aircraft from their component parts you can repel the alien invaders. Such a tedious theme can often be the basis of a stunning arcade game. However in this case I would be quite willing to hang up my joystick and surrender.

In **Chopper Squad,** from Interceptor Software, you are the pilot of a rocket helicopter, that's the red smudge in the middle of the screen. Also on the screen are three horizontal lines. They are the helicopter landing pads onto which will fall jet fighter parts. But where the parts are falling from I don't know. Maybe someone up there doesn't like aliens either.

Painfully slowly you fly your helicopter around the screen and pick up one of the components.

You then move back to the bottom of the screen and fly through the bottom right hand corner. At this point the jet part vanishes and reappears in the score window at the top of the screen.

As each component is successfully retrieved the fighter is constructed.

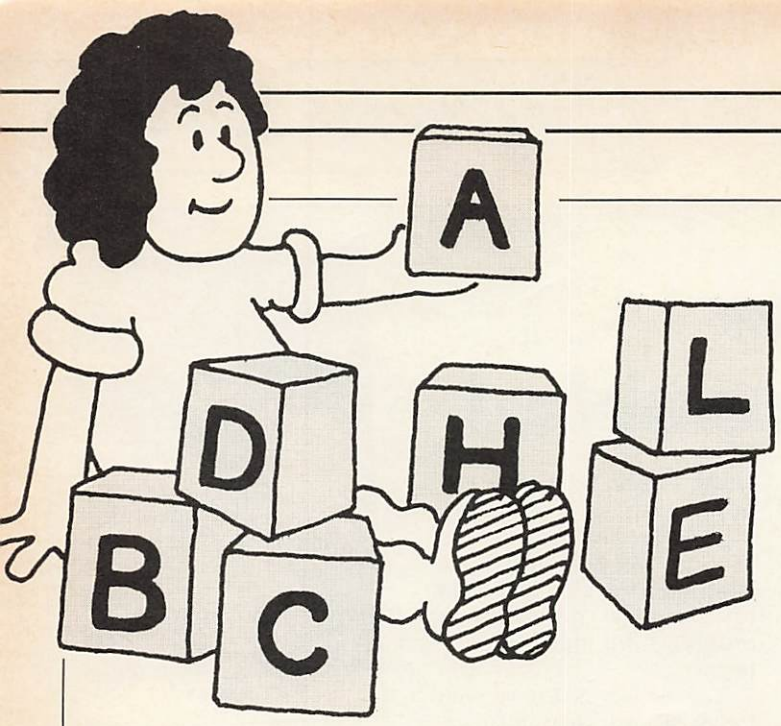
While you are merrily employed in the aircraft construction business a swarm of blue flying saucers drift aimlessly around the screen. On early levels they present no challenge, as they are easily despatched with one blast of your laser cannon.

Having built your first plane the screen clears and the flying saucers are replaced by red fruit gums. These, too, provide very little opposition.

On the third screen however the red hot dogs decide to turn nasty and follow your every move relentlessly. These McDonald's Specials take a great deal of shaking off when your top speed is almost in excess of five miles per hour.

Should you be looking for a game that might just test your granny's reflexes to the full then this is just what you were looking for.

James Riddell



Now for some more registers

In Part V of his introduction to Machine Code MIKE BIBBY reveals the secrets of the remaining Z80 registers

SO far in this series we've concentrated on the A register, a single byte internal memory location within the Z80 processor itself. However, I have hinted darkly that there are other registers available, and we'll be looking at some of them this month.

The registers we're interested in are the B, C, D, E, H and L registers. Each of these, like the A register, can store a single byte.

There are other registers, such as the F register, but for the moment we'll just concentrate on the half dozen I've listed.

(Incidentally, there are no G, I, J or K registers.)

Since they are single byte, or eight bit registers, we can load them directly with a number in the range 0 to 255, just as we did register A.

That is, commands such as:

```
LD B,&2A
```

and

```
LD H,&0C
```

exist.

We can represent this type of operation, where we load a register with a byte, symbolically as:

```
LD r,n
```

where: **LD** *stands for LoaD.*
r *stands for one of the registers A, B, C, D, E, H, L.*
n *stands for a number in the range 0 to 255. We refer to such numbers as constants.*

Table I shows the opcodes for each of these register loads, including LD A,n which we're already familiar with.

From Table I you can see that:

```
LD B,&2A
```

would translate into:

```
06 2A
```

whereas:

```
LD H,&0C
```

would give:

```
26 0C
```

Right, so we now know how we can load the various registers with numbers – but why would we want to do so?

Well, you'll know from your programming in Basic how valuable variables are for keeping track of things. In machine code we haven't got that luxury. However, we can use the registers as rudimentary variables.

Even though each single register can only hold a number in the range 0 to 255, you can still achieve some powerful results. After all, even Basic was written in machine code!

LD B,n	06	n
LD C,n	0E	n
LD D,n	16	n
LD E,n	1E	n
LD H,n	26	n
LD L,n	2E	n
LD A,n	3E	n

Table I: LD r,n opcodes

As an example of the sort of thing I mean, let's locate the text cursor at a given column and row on the screen and then print an asterisk there.

Fortunately there's a firmware routine that allows us to do just that – it's located at &BB75. I'm going to call it PosTCur, short for Position Text Cursor.

To use it, we load the H register with the column we want and the L register with the row we want. We then call PosTCur.

Having positioned our cursor, we still have to print the asterisk. To do this we simply load the A register with the Ascii code for asterisk (&2A) then call our CharOut routine at &BB5A to print it out.

While we're christening routines, let's call the routine at &BB6C – which clears the text screen – ClrText. We use it in the program below before positioning our cursor, printing the asterisk and then returning to Basic.

```
address   hex code    mnemonics
 3000     CD 6C BB    CALL ClrText
 3003     26 14       LD H,&14
 3005     2E 0C       LD L,&0C
 3007     CD 75 BB    CALL PosTCur
 300A     3E 2A       LD A,&2A
 300C     CD 5A BB    CALL CharOut
 300F     C9          RET
```

In this program the H and L registers have clearly been used as variables to specify where we want our text cursor positioning. Try altering the values you give H and L to place the cursor in different positions on the screen.

Of course, as I've said, we can likewise load the other registers with numbers in the range 0 to 255. The simplest way to demonstrate this would be to load the register in question with a number and then to poke it into Hexer's "workspace" (&2FF8 – &2FFF), where we could examine it at leisure.

As you'll recall from last month, this works well enough for the A register:

```
LD A,n
LD (&2FF8), A
RET
```

Here you load A with the number you require (shown here as n – an eight bit number) and then load memory location &2FF8 with the contents of A. In effect, you're poking the A register's contents into &2FF8.

You'd then look at &2FF8 to see that it really does contain what you'd loaded A with.

Unfortunately, this doesn't work with the other registers, because you can't poke into memory with a register other than the A register. So:

```
LD B,n
LD (&2FF8),B
RET
```

fails because there is no such instruction as:

```
LD (&2FF8),B
```

More formally,

```
LD (pq),r
```

does not exist except where r is the A register. Here pq refers to a two byte, 16 bit number. Remember, we're specifying a memory address inside those brackets, so we need two bytes.

Similarly, you can't peek with registers other than A, so:

```
LD B,(&2FF8)
```

is also illegal. More formally,

```
LD A,(pq)
```

is the only peek allowed.

What we can do, though, is to adopt a roundabout approach, loading the number into the register under examination, then loading the A register with the contents of that register and poking the A register into memory.

The code would be something like:

```
LD B,n
LD A,B
LD (&2FF8),A
RET
```

It relies on a new instruction:

```
LD A,B
```

This allows us to load register A with the contents of register B. In fact, this is just one of a whole set of opcodes that allow us to transfer the contents of one register to another. We describe this class of opcodes as:

```
LD r,r'
```

where both r and r' are any of the registers A, B, C, D, E, H, L;

Table II shows the opcodes – there are quite a few!

When you're using Table II, remember that, as it's set out, the first register (r) specifies the row and the second register (r') gives the column it's in. Hence the opcode for LD B,C is 41.

You'll probably have noticed a pattern. If not, it might help you to see it if I tell you that there's both a row and a column missing between the L and A registers. We'll see what that means later.

You'll see such a pattern in many of our opcodes – it helps the Z80 to work out what the opcode's telling it to do.

By the way, LD r,r' copies the contents of r' into r. In doing so, it doesn't alter the value in r', so when the micro's finished the opcode, both r and r' will have the same value.

To demonstrate some of these opcodes in action, we can play a game of "Chinese Whispers" with our registers. Let's load H with &2A, then pass it on from register to register until we reach the A register, when we'll print it out with CharOut.

address	hex code	mnemonics
3000	26 2A	LD H,&2A
3002	6C	LD L,H
3003	5D	LD E,L
3004	53	LD D,E
3005	4A	LD C,D
3006	41	LD B,C
3007	78	LD A,B
3008	CD 5A BB	CALL CharOut
300B	C9	RET

So far we've learned to do quite a bit with our registers: we can load them directly with numbers, and we can copy numbers from one register to another.

There are other operations we can perform on the registers. For instance, we can INC and DEC them. INC A as you'll recall from last month, increases the value of the A register by one. DEC A decreases its value by one.

```
INC r
```

and

```
DEC r
```

where r is any of registers B, C, D, E, H, L, have similar effects on the specified register.

Table III contains the opcodes you need for INCrementing and DECrementing our eight bit registers. As we found out last month, none of them affect the carry flag.

To illustrate their use, we can incorporate them into our previous code, to place not one but two asterisks on the screen, the second directly below the first.

Since L contains the row you might think that the way to do it would be to run our code as normal up to printing the first asterisk, then INC L, to give us the row below (think!), call

		r'						
		B	C	D	E	H	L	A
r	B	40	41	42	43	44	45	47
	C	48	49	4A	4B	4C	4D	4F
	D	50	51	52	53	54	55	57
	E	58	59	5A	5B	5C	5D	5F
	H	60	61	62	63	64	65	67
	L	68	69	6A	6B	6C	6D	6F
	A	78	79	7A	7B	7C	7D	7F

Table II: Opcodes for LD r,r'

r	INC r	DEC r
B	04	05
C	0C	0D
D	14	15
E	1C	1D
H	24	25
L	2C	2D
A	3C	3D

Table III: INC r and DEC r opcodes

PosTCur again, then print another asterisk, as in:

```
CALL ClrText
LD H,&12
LD L,&0C
CALL PosTCur
LD A,&2A
CALL CharOut
INC L
CALL PosTCur
CALL CharOut
RET
```

Unfortunately this won't work, as calling PosTCur actually changes the values that the H, L, A registers contain. This isn't surprising since PosTCur is just another machine code routine and will need "variables" itself.

It's inconvenient for us, though. It would be nice if L still had the same value so we could just INC it to get the cursor on the next row when we call PosTCur.

Nor can we guarantee that H still contains the correct column or that the second call to CharOut will print an asterisk – the value in A will have changed.

When a machine code routine changes registers, we say it corrupts the registers. When a routine leaves the registers intact we say it preserves the registers.

If you find your machine code programs going wrong for no obvious reason, check to see if you're wrongly assuming that some of your CALLs to other subroutines are preserving the registers.

In practice you'll tend to find some registers preserved and some not. The two routines we've relied on most heavily, CharOut and CharIn, preserve all the registers, fortunately.

Even PosTCur leaves some registers unchanged – the B, C, D, E registers. We can take advantage of this by transferring our "sensitive" register values (A, H, L) into these unchanged registers for safe keeping, transferring them back when we need them.

By using these ideas, we can successfully recode our previous routine to allow us to place two asterisks on the screen, one directly below the other.

We've used B to store H, C to store L and D to store A. Later we'll discover easier ways of getting round the problems of corrupt registers. As it is, the above example certainly illustrates operations with our eight bit registers.

```
address  hex code   mnemonics
 3000    CD 6C BB   CALL ClrText
 3003    26 12      LD H,&12
 3005    44         LD B,H
 3006    2E 0C      LD L,&0C
 3008    4D         LD C,L
 3009    CD 75 BB   CALL PosTCur
 300C    3E 2A      LD A,&2A
 300E    57         LD D,A
 300F    CD 5A BB   CALL CharOut
 3012    60         LD H,B
 3013    69         LD L,C
 3014    2C         INC L
 3015    CD 75 BB   CALL PosTCur
 3018    7A         LD A,D
 3019    CD 5A BB   CALL CharOut
 301C    C9         RET
```

By working out what's happening you'll definitely master the ideas. Why not try printing out a column of three asterisks? B, C, D will still be preserved, remember.

Last month we saw that we could add and subtract numbers from the A register. You might be tempted to think that since we could INC and DEC the other registers we could ADD and SUB them also. Unfortunately not:

```
ADD B,B
```

and

```
SUB C,4
```

do not exist!

You can, however, ADD the contents of a register to the A register.

Similarly you can SUB the contents of a register from the A register.

In both cases, the answer is left in the A register.

Formally,

```
ADD A,r
```

and

```
SUB r
```

exist, where r is any register of B, C, D, E, H, L, A.

Notice, by the way, that when you're adding, you have to specify A followed by the register you're adding to it. When you're subtracting you only need to specify the register you're subtracting – the micro automatically knows you're taking it from the A register.

r	ADD A,r	SUB r
B	80	90
C	81	91
D	82	92
E	83	93
H	84	94
L	85	95
A	87	97

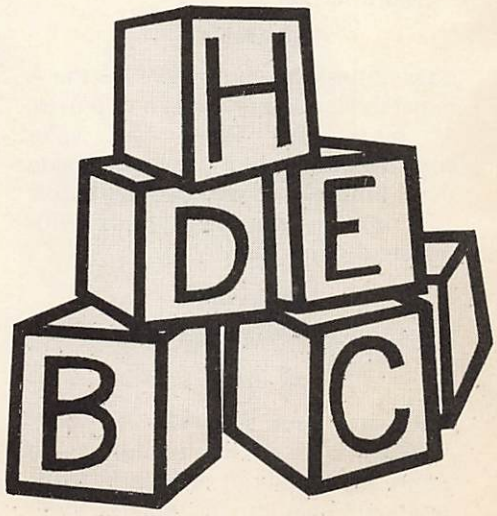
Table IV: ADD A,r and SUB r opcodes

Incidentally, the way the answers end up, or accumulate, in the A register explains its alternative name; the accumulator. And, of course, since the answers do end up in the A register, they're handy for poking into memory. (Remember we do our eight bit pokes into memory via the A register.)

The opcodes for ADD and SUB are given in Table IV. You'll need them to do the following sums, Problems I to III. Don't try them before you've read the whole of the article, though, as in a moment I'll give you what my maths teacher used to call a worked example.

The answers to each of the sums is sorted in &2FF8, so you can examine them easily with option 2 of Hexer. You should be able to follow what's happening without too much trouble.

Remember, if you add 1 to &FF the answer is 0 when you're dealing with eight bit numbers. Similarly, if you take 1 from 0, the answer is &FF. As



Problem I

```
LD B,&01
LD A,&01
ADD A,B
LD (&2FF8),A
RET
```

Problem II

```
LD C,&1
LD A,&F
SUB C
LD (&2FF8),A
RET
```

Problem III

```
LD D,&FF
LD A,&2
ADD A,D
LD (&2FF8),A
RET
```

When a machine code routine changes registers, we say it corrupts the registers. When a routine leaves them intact we say it preserves the registers.

we saw last month, the numbers cycle round.

Notice that I haven't spelt out the actual hex opcodes for you – you'll have to do that yourself. This is called hand assembling the code. Tables I to V give all the opcodes you need, Table V being a resumé of the codes we met last month.

A few hints on hand assembling might be useful here:

- Write out your code on paper before entering it into Hexer.
- Make sure you're dealing with hex numbers – any decimal numbers will have to be translated.
- Check that the code ends with the opcode for RET, &C9.
- Be careful with your writing – it's all too easy to confuse 8 with B, 2 with 7, and E with F.
- Make certain that you write addresses in lo byte, hi byte form. That is, &2FF8 becomes F8 followed by 2F in a machine code program.
- Double check the code before you run it!

Now for the worked example.

mnemonic	opcode
CALL pq	CD q p
RET pq	C9
LD A,(pq)	3A q p
LD (pq),A	32 q p

Table V: Additional opcodes

Suppose you had to hand assemble:

```
LD A,8
SUB A
LD (&2FF8),A
RET
```

Firstly you'd decide where you were putting it in memory. Since we're using Hexer, we tend to put everything at &3000.

Look at the first instruction:

```
LD A,8
```

we can see that it's of the LD A,n type. As Table I shows us, the opcode for this is 3E, followed by the number we want loading in. In this case it's 8, and there are no translation problems since it's the same in hex and decimal. So the first line translates as:

```
3E 08
```

The next line for us to code is:

```
SUB A
```

This is a rather interesting piece of code, since it subtracts the A register from the A register! I think you can guess the result . . .

From Table IV, you can see that the code for this is 97, so we can add this to our string of bytes in memory to give:

```
3E 08 97
```

The next instruction,

```
LD (&2FF8),A
```

is the poke type instruction we saw last month. Its opcode is 32. The trick is to make sure we follow it with the address we want to poke into split into lo byte, hi byte order. Given this, our code becomes:

```
3E 08 97 32 F8 2F
```

All that remains now is to translate the last instruction,

```
RET
```

The opcode for RET is C9 – you should really know this by heart now. Our code now becomes:

```
3E 08 97 32 F8 2F C9
```

which, once we've done all our checks, we can enter via Hexer.

I tend to use a slight variation on this technique, writing the code for each instruction on a separate line. I also label each line with the memory location that the first byte (the opcode) of that line is stored in. Doing things this way makes it easier to compare the code with the original mnemonics.

Also, knowing the starting address of each instruction will be handy later on when we do more with jumps and so on.

Doing things this way, the above code would be:

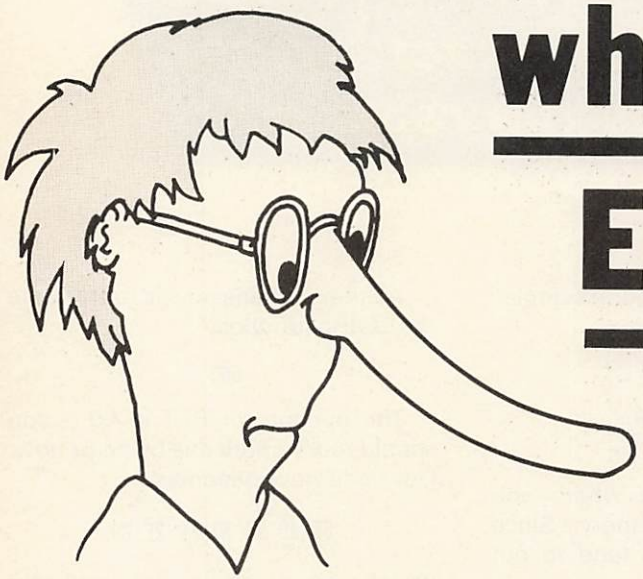
```
3000    3E 08
3002    97
3003    32 F8 2F
3006    C9
```

That's all for this month. You may have noticed that, apart from specifying addresses, the numbers we've been dealing with have all been one byte long.

Next month we'll have a look at pairing up the registers to allow us to handle two byte numbers.

A final tip: You might find it useful to copy the tables of opcodes onto card for easy reference – that's what I do.

EOR–A way to find out who's telling the Exclusive truth



MIKE BIBBY continues his series on the fundamental workings of the CPC464

IN the last article we looked at the AND and OR operations on binary numbers – logical operations, as they are known. These were simply rules for combining numbers bit by bit. We shall continue our exploration this month with a look at the EOR operation.

EOR stands for Exclusive OR – sometimes people call it XOR. Either way it's the same thing. EOR is a variant on the way we normally use the term OR.

For example, if I say:

Mike OR Pete wears glasses

this is true if Mike wears glasses, OR Pete wears glasses, OR both Mike *and* Pete wear glasses.

Now it's this last case of OR we're interested in, where they both wear glasses. EOR works just like OR up to this point. However, EOR does not "allow" both of them to wear glasses. Either one does, or the other, but not both.

To put it another way, the one who wears the glasses does so *exclusively*.

If both are wearing glasses then while:

Mike OR Pete wears glasses

would be true,

Mike EOR Pete wears glasses

would be a downright lie!

We could signify that a statement is true with the letter T, and use F for false. At school our teachers used ticks for truth and crosses for false. Since we're using computers, though, we'll use numbers: 1 will denote true and 0 will denote false. We've chosen 1 and 0 because they fit in so well with the binary system.

So, in the above example, if Mike has glasses we can give Mike the value 1. If Pete hasn't glasses we can give Pete the value 0. Table I shows the idea, applied to each combination of spectacle user. The ones and zeros are known as truth values, states or conditions.

As you can see, there are four possible cases as far as Mike and Pete wearing glasses are concerned: neither can wear them as in case 1, where both Mike and Pete has 0 value. Then again, Pete may wear them (1) whereas Mike does not (0), case 2, and so on.

If you look carefully at the numbers involved in all four cases, you see that we've got four pairs of bits we can combine. Each pair of bits is made up of the "truth bit" for Mike and the "truth bit" for Pete.

What I've done in Table II is to combine these pairs for all four cases in accordance with our OR rules. We've stored the result in a third column.

We call such a table a Truth Table. In this case, it's the truth table for OR.

	Wears glasses		
	Mike	Pete	
Case 1	0	0	neither wears glasses
Case 2	0	1	Pete wears glasses
Case 3	1	0	Mike wears glasses
Case 4	1	1	Both wear glasses

Table I

Mike wears glasses	Pete wears glasses	Mike OR Pete wears glasses
0	0	0
0	1	1
1	0	1
1	1	1

Table II

We can use it to work out the result for any OR combination of two bits. All we have to do is to find the row that starts with the two bit values we're combining and then look in the third column for the result.

Table III shows a similar table for:

Mike AND Pete wear glasses

Again the first two columns are identical, covering all four possible cases. The third column combines them according to the AND rules.

Mike wears glasses	Pete wears glasses	Mike AND Pete wear glasses
0	0	0
0	1	0
1	0	0
1	1	1

Table III

Look again at Table II. This corresponds in a sense to our binary rule for OR: you get a 1 if either or both bits you combine contain a 1.

However if when talking about Mike and Pete you mean OR in the exclusive sense, EOR, then the combination of Mike wearing glasses and Pete also wearing glasses would have to be false. This is because EOR means either one or the other wears glasses, but not both – it's *exclusively* one or the other.

If we do mean EOR in this exclusive sense we'd write our statement about them as:

Mike EOR Pete wears glasses

Its Truth table is given in Table IV:

Mike wears glasses	Pete Wears glasses	Mike EOR Pete wears glasses
0	0	0
0	1	1
1	0	1
1	1	0

Table IV

If you look at each case, you'll see that the only time Mike EOR Pete is true is when either one or the other wears glasses, but not both (or neither).

More formally, if both bits are 0, or both bits are 1 the result is 0. If either is 1 and the other is 0 the result is 1. To put it another way, if the bits are identical the result is 0, otherwise the result is 1.

Let's have a look at how we EOR binary pairs of numbers. It's the same as for OR and AND – just apply the rules for EORing to each pair of bits in succession. For example:

```
        10110110
EOR     11100101
gives   01010011
```

Take a look at what happens when you EOR a number with zero:

```
        10110110
EOR     00000000
gives   10110110
```

that is, when you EOR a number with zero it leaves that number unchanged. Also something interesting happens when you EOR a number with itself:

```
        10110110
EOR     10110110
gives   00000000
```

Whenever you EOR a number with itself, the result is zero. This is as it should be: remember, when you EOR two identical bits the result is zero.

Now EOR has a property which makes it quite useful – let's look what happens when we take a number, EOR it with a second number and then go on to EOR the result once more with that second number.

```
First number          10101101
Second number   EOR   01101000
Result                11000101
Second number   EOR   01101000
Final result          10101101
```

As you can see, the first number has magically re-appeared! This always happens when you EOR twice with the same number as, in a sense, the two EORings cancel each other out.

Table V summarises the process for all four possible pairs of one-bit numbers. As you can see, for all the cases the final resulting bit (when the first bit has been EORed twice with the second) is identical to the first bit.

Another way to think of it is that we are doing:

first number EOR second number EOR second number

Taking the underlined part first, we've already seen that any number EORed with itself gives a zero result. So what we're really doing is:

first number EOR 0

which, as we've also seen, must leave just the first number, since EORing with zero leaves a number unchanged.

All this may seem rather abstruse, but actually it's quite useful. In fact we tend to use AND, OR and EOR quite often in graphics, particularly in animation.

To simulate movement we frequently print something on the screen, then after leaving it there for a while to register on the eye, we blank it out and print it in a new position and so on.

Sometimes we blank the character out by printing it again in the same place but in the background colour.

We can, however, use EOR. If we use EOR to place our character on the screen – never mind exactly how for the moment – when it comes to wanting rid of it, we can just repeat ourselves.

That is, we just EOR the character on again. As we've seen, the effect of two EORs is to cancel each other out. In this case, they cancel out to the original background – and the character disappears.

Don't worry too much about the details, I just want to convey the general idea. Our graphics series, and Kevin Edward's articles on animation will take it further.

The point is, logical operators, as AND, OR and EOR are known, can be invaluable to both the Basic and machine code programmer.

● *Next month we'll conclude our series with a brief look at the idea of masks.*

First bit	Second bit	Result 1st EOR	Second bit again	Result 2nd EOR
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1
1	1	0	1	1

Table V

Ready Reference: String handling

Get the facts at your fingertips with the fifth of our Amstrad reference charts

LEFT$ slices off part of a string. As you might guess, it returns the leftmost part.

```
PRINT LEFT$("remainder",6)
```

results in "remain" appearing on the screen. **LEFT$** has taken the leftmost six characters from "remainder", ignoring the rest. Try:

```
word$="thisbitthatbit"
PRINT LEFT$(word$,7)
```

and see what you get. The number after the string to be sliced decides how many characters are taken. What if this number is greater than the number of characters in the string to be sliced?

```
PRINT LEFT$("four",5)
```

shows that only the four characters are produced. In effect, the string is left intact.

RIGHT$** follows on from **LEFT$, slicing off the righthand part of the string.

```
word$="leftrights"
PRINT RIGHT$(word$,6)
```

The number inside the brackets specifies the character at which the slicing is to start – measured inwards from the end of the string to be sliced. It is the number of characters to be lifted from the righthand part of the original string.

```
PRINT RIGHT$("12345678",2)
```

If the number specified is greater than the length of the string, the string remains intact.

```
PRINT RIGHT$("five",6)
```

STRING$ allows you to create long strings made up of single characters. If you want a row of pluses you'd use:

```
PRINT STRING$(20,"+")
```

Notice that it only works with single characters.

```
character$=STRING$(10,"ab")
PRINT character$
```

SPACE$ allows you to create long lines of spaces. You'll see that:

```
PRINT SPACE$(11)
```

is a lot clearer than:

```
PRINT "          "
```

Although you can't see them, the spaces are there.

```
seeing$="is"+SPACE$(12)+"believing"
PRINT seeing$
```

MID$** is, in effect, a combined **RIGHT$ and **LEFT$**. With it you can not only decide where the slice is to start, but also how characters of the original string are to be taken.

```
PRINT MID$("level",2,3)
```

should leave you with "eve" while:

```
PRINT MID$("madam",2,4)
```

should produce "adam".

The first number following the specified string indicates where the slice is to begin. The second number decides how long that slice will be.

```
word$="level":nextword$="ning all"
combined$=MID$(word$,2,3)+nextword$
PRINT combined$
```

If the final figure is omitted you just get the rest of the string from the start character.

```
PRINT MID$("rotor",2)
```

gives "otor".

If the character specified is greater than the number of characters in the string to be sliced then you get nothing.

```
slice$=MID$("123456",7,3)
PRINT LEN(slice$)
```

If you ask for more characters than the original string can supply, the micro does the best it can.

```
PRINT MID$("123456",3,5)
```

Escape from the CASTLE OF FEAR

WHILE enjoying a fine meal with a friendly dwarf you find yourself transported to the guest room of a large castle.

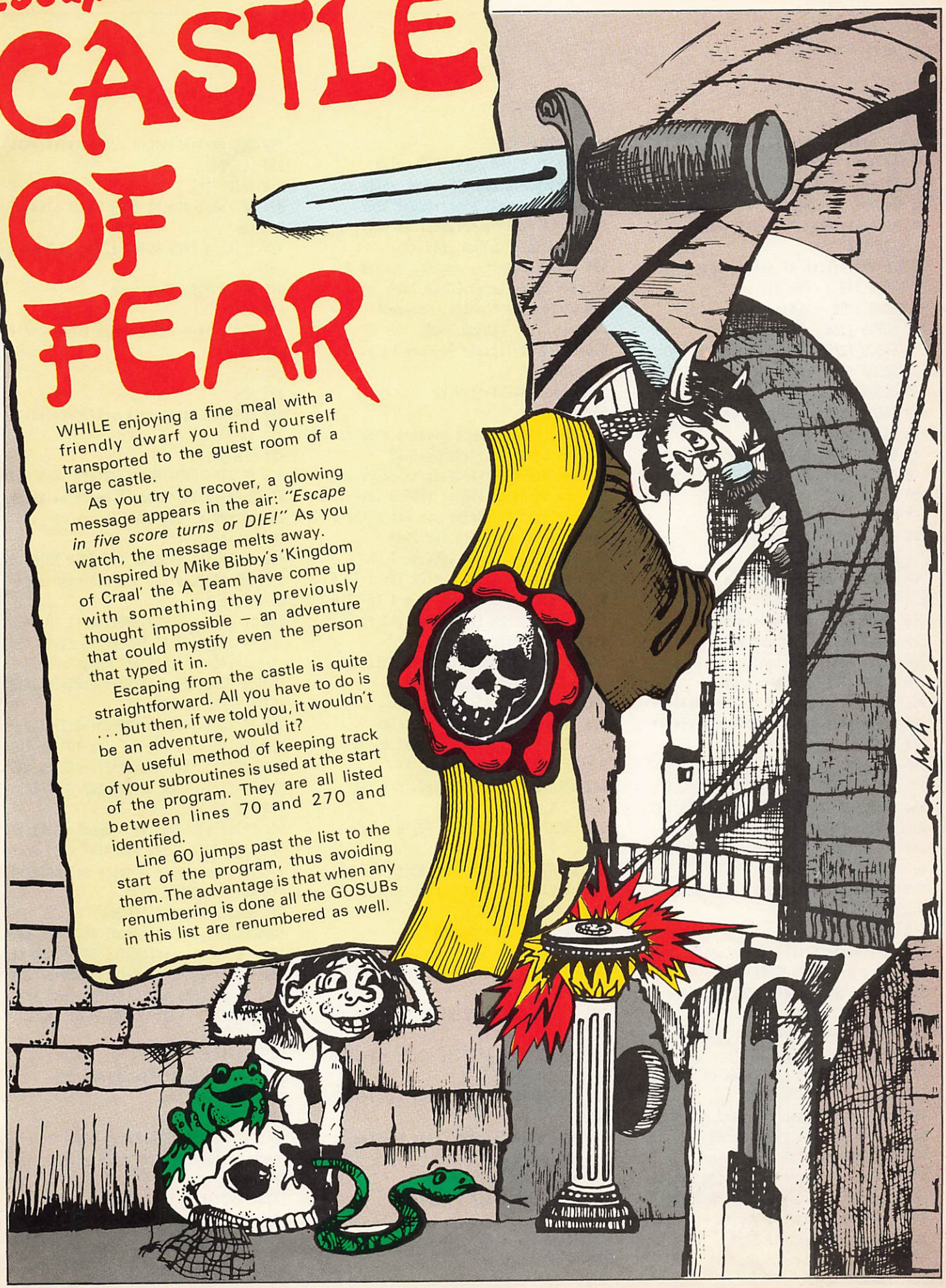
As you try to recover, a glowing message appears in the air: *"Escape in five score turns or DIE!"* As you watch, the message melts away.

Inspired by Mike Bibby's 'Kingdom of Craal' the A Team have come up with something they previously thought impossible – an adventure that could mystify even the person that typed it in.

Escaping from the castle is quite straightforward. All you have to do is . . . but then, if we told you, it wouldn't be an adventure, would it?

A useful method of keeping track of your subroutines is used at the start of the program. They are all listed between lines 70 and 270 and identified.

Line 60 jumps past the list to the start of the program, thus avoiding them. The advantage is that when any renumbering is done all the GOSUBs in this list are renumbered as well.



```
10 REM **************************
20 REM ***   Castle Of Fear   ***
30 REM *** (C) Computing With ***
40 REM ***     The Amstrad    ***
50 REM **************************
60 GOTO 280
70 REM ******list of subroutines*****
80 GOSUB 900    REM player
90 GOSUB 1180   REM help
100 GOSUB 1250   REM waiting
110 GOSUB 1270   REM move troll
120 GOSUB 1340   REM attack
130 GOSUB 1400   REM you move
140 GOSUB 1440   REM move jewels
150 GOSUB 1550  REM troll
160 GOSUB 1630  REM inventory
170 GOSUB 1650  REM enter
180 GOSUB 1850  REM directions
190 GOSUB 1970  REM get
200 GOSUB 2050  REM drop
210 GOSUB 2190  REM find
220 GOSUB 2260  REM contents
230 GOSUB 2340  REM quit
240 GOSUB 2390  REM messages
250 GOSUB 2490  REM initialise
260 GOSUB 2680  REM centre text
270 GOSUB 3380  REM short delay
280 REM *******program starts*********
290 REM*******************************
300 MODE 1
310 BORDER 0
320 INK 0,15:INK 1,0:INK 2,3:INK 3,1
330 c%=1:y%=3:msg$="CASTLE OF FEAR":GO
SUB 2680: REM Centre Text
340 y%=4:msg$="by Mark Smiddy and Mich
ael Noels":GOSUB 2680
350 y%=5:msg$="adapted by Alan McLachl
an.":GOSUB 2680
360 c%=2
370 y%=7:msg$="Expert adventuring is b
ecoming a pastime":GOSUB 2680
380 y%=8:msg$="for the idle rich these
 days.":GOSUB 2680
390 y%=9:msg$="Long ago Kings would pa
y high prices":GOSUB 2680
400 y%=10:msg$="to the likes of you to
 rescue their":GOSUB 2680
410 y%=11:msg$="beautiful,and not-so-b
eautiful daughters":GOSUB 2680
420 d%=30:GOSUB 3380: REM delay
430 c%=3:y%=13:msg$="Not so now.Even t
he dragons have stopped":GOSUB 2680
440 y%=14:msg$="eating elvies - in fac
t Adventureland":GOSUB 2680
450 y%=15:msg$="is not what it used to
 be.":GOSUB 2680
460 d%=30: GOSUB 3380: REM delay
470 c%=2:y%=17:msg$="However while enj
oying a fine meal with":GOSUB 2680
480 y%=18:msg$="a friendly dwarf, you
 find yourself":GOSUB 2680
490 y%=19:msg$="transported to the gue
st room of a large":GOSUB 2680
500 y%=20:msg$="castle. As you try to
 recover....." :GOSUB 2680
510 y%=24:c%=1:msg$="Any key to go on.
":GOSUB 2680
520 WHILE INKEY$="":WEND
530 CLS
540 c%=2:y%=3:msg$="a glowing message
 appears in the air":GOSUB 2680
550 INK 3,6:c%=3:y%=6:msg$="Escape in
 5 score turns or DIE!!!":GOSUB 2680
560 c%=2:y%=9:msg$="As you watch the m
essage melts away.":GOSUB 2680
570 FOR delay = 1 TO 1000:NEXT
580  WINDOW #1,1,40,5,7:FOR l%= 1 TO 4
0::FOR k%=1 TO 3:FOR delay = 1 TO 20:N
EXT:INK 1,15:PEN 1:LOCATE #1,l%,k%:PRI
NT#1, CHR$(143);:NEXT:NEXT
590 y%=12:c%=1:msg$="Use the following
 keys to play.":GOSUB 2680
600 WINDOW #2,6,35,13,22
610 INK 0,15:INK 1,0:INK 2,3:INK 3,1
620 PEN 3:LOCATE#2, 3,2:PRINT #2,"N =
North        S = South"
630 LOCATE #2,3,4:PRINT #2,"E = East
       W = West"
640 LOCATE #2,3,6:PRINT #2,"U = up
       D = Down"
650 LOCATE #2,12,8:PRINT #2,"Q = Quit"
660 y%=23:PEN 3:msg$="Press any key to
 go on":GOSUB 2680
670 WHILE INKEY$="":WEND
680 CLS #2
690 LOCATE #2,3,2:PRINT#2,"@ = Look
      A = Attack"
700 LOCATE #2,3,4:PRINT #2,"G = Get
       L = Drop"
710 LOCATE #2,3,6:PRINT #2,"P = Wait t
urn    H = Help"
720 LOCATE #2,9,8:PRINT #2,"I = Invent
ory"
730 y%=23:PEN 3:msg$="Press any key to
 go on":GOSUB 2680
740 WHILE INKEY$="":WEND
750 GOSUB 2490: REM initialise
760 REM
770 REM
780 REM
790 REM *********MAIN LOOP***********
800 CLS #4
810 WHILE 1=1
820 IF room%<>last% THEN GOSUB 1700:RE
M enter
830 IF W%=0 THEN GOSUB 1270:REM move t
roll
840 GOSUB 900:REM player
850 WEND
860 REM *******END OF MAIN LOOP*******
870 REM
880 REM
890 REM
900 REM***********player*************
910 W%=0
920 PEN #5,3
930 IF (INT(RND*100)+1)-T%<10 THEN PRI
NT #5:PRINT #5,"A small dwarf walks up
 to you":PRINT #5,"says 'Hurry up' and
 then walks off."
940 IF INT(RND*100)+1 <10 THEN PRINT #
5:PRINT #5,"Listen, you can hear stran
ge voices."
950 IF C%>0 AND INT(RND*100)+1 <8 THEN
 PRINT #5:PRINT #5,"A sprite takes the
 ";coin$(C%);" coin and runs.":C%=C%-1
:WHILE R%=14 OR cts%(R%)<>0:R%=INT(RND
*27)+1:WEND:cts%(R%)=1
960 IF (coin%(1)<>0 AND coin%(2))<>0 A
ND coin%(3)<> 0 AND NOT (Y% XOR 0) THE
N IF ABS(coin%(3)-coin%(2))=ABS(coin%(
2)-coin%(1)) THEN N%=7:GOSUB 2390:END
970 PEN #4,1
980 PRINT #4:PRINT #4,"Turns - ";T%:IF
 T%>=100 THEN PRINT #4:PRINT #4,"'Sorr
y time's up says a ghostly voice":N%=I
NT(RND*4)+1:GOSUB 2390
990 WHILE INKEY$<>"":WEND
1000 PEN #4,2:G$="":PRINT #4:PRINT #4,
"Your move  ";:tt=1:WHILE G$="" AND tt
<4000:G$=INKEY$:tt=tt+1:WEND:G$=UPPER$
(G$):IF G$=CHR$(13) OR G$="" THEN G$="
w":PRINT #4, "Wait" ELSE PRINT #4, G$
1010 RESTORE 3260
1020 des$=""
1030  WHILE des$<>G$ AND N%<>255
1040 READ des$,N%
1050  WEND
1060 IF N%=255 THEN PRINT #4,"I do not
 understand":PRINT #4,"Try another key
":RETURN
1070 T%=T%+1
1080 IF N%<=6 THEN GOSUB 1400:RETURN:
REM move direction
1090 IF N%=7 THEN GOSUB 1700:T%=T%-1:R
```

```
ETURN: REM look
1100 IF N%=8 THEN GOSUB 1970:RETURN: R
EM get article
1110 IF N%=9 THEN GOSUB 2050:RETURN: R
EM drop article
1120 IF N%=10 THEN GOSUB 1630:RETURN:
REM inventory
1130 IF N%=11 THEN GOSUB 2340:RETURN:
REM quit
1140 IF N%=12 THEN GOSUB 1250:RETURN:
REM wait
1150 IF N%=13 THEN GOSUB 1340:RETURN:
REM attack
1160 IF N%=14 THEN GOSUB 1180:RETURN:
REM help
1170 RETURN
1180 REM *********help*************
1190 RESTORE 3360
1200 WHILE S%<T% AND S%<>95
1210 READ S%,des$
1220 WEND
1230 PRINT des$
1240 RETURN
1250 REM*******waiting***************
1260 PRINT #4,"You wait...Time passes"
;:FOR N=1 TO 10:PRINT #4,".";:FOR dela
y=1 TO 150:NEXT:NEXT:RETURN
1270 REM**********move troll**********
1280 tr%=troll%(1):W%=1
1290 GOSUB 1440:GOSUB 1550 REM move je
wel/create troll
1300 IF troll%(1)=room% AND tr%<>room%
 THEN PRINT #4,"A large troll enters,
holding";:IF troll%(2)=0 THEN PRINT #4
," nothing."
1310 IF troll%(1)=room% AND tr%<>room%
 AND troll%(2)<>0 THEN PRINT #4,"a gli
ttering jewel."
1320 IF troll%(1)<>room% AND tr%=room%
 THEN PRINT #4,"A large troll leaves."
1330 RETURN
1340 REM***********attack*************
1350 IF cts%(room%)=0 THEN PRINT #4,"A
ttacking walls may be seen as insanity
 and has no effect."
1360 IF (cts%(room%) AND 1)=1 OR (cts%
(room%) AND 2)=2 THEN PRINT #4,"Your b
lows bounce cleanly of the pedestal"
1370 IF (cts%(room%) AND 4)=4 THEN PRI
NT #4,"The troll just giggles !"
1380 IF (cts%(room%) AND 8)=8 THEN PRI
NT #4,"The angry troll throws a large
jewel.":N%=INT(RND*4)+1:GOSUB 2390
1390 RETURN
1400 REM ************move************
1410 IF dir%(N%)=0 THEN PRINT #4,"Can'
t go that way.":RETURN
1420 room%=dir%(N%)
1430 RETURN
1440 REM *********move jewels********
1450 IF (jew%(1)<>0 AND jew%(2)<>0 AND
 jew%(3)<>0) AND NOT (X% XOR 0) THEN I
F ABS(jew%(3)-jew%(2))=ABS(jew%(2)-jew
%(1)) THEN N%=INT(RND*4)+1:GOSUB 2390
1460 R%=INT(RND*3)+1
1470 IF R%<=2 THEN troll%(1)=jew%(INT(
RND*3)+1)
1480 IF R%=3 THEN troll%(1)=jew%(INT(R
ND*3)+1)
1490 IF troll%(1)=0 THEN troll%(1)=INT
(RND*27)+1
1500 GOSUB 1550: REM troll
1510 PEN #4,3
1520 IF troll%(2)>0 AND (cts%(troll%(1
)) AND 3)=0 THEN cts%(troll%(1))=cts%(
troll%(1)) OR 2:jew%(troll%(2)-1)=trol
l%(1):troll%(2)=troll%(2)-1:PRINT #4:P
RINT #4,"You hear a distant WHOOOF!!":
RETURN
1530 IF troll%(2)<4 AND (cts%(troll%(1
))AND 2)=2 THEN cts%(troll%(1))=cts%(t
roll%(1)) XOR 2:jew%(troll%(2))=0:trol
l%(2)=troll%(2)+1:PRINT #4:PRINT #4,"Y
ou hear a distant SLUUURP!!"
1540 RETURN
1550 REM********troll**************
1560 FOR N%=1 TO 3
1570 IF jew%(N%)=troll%(1) THEN cts%(t
roll%(1))=cts%(troll%(1) OR 2)
1580 NEXT
1590 cts%(room%)=cts%(room%) AND 243
1600 IF room%=troll%(1) AND troll%(2)
<> 0 THEN cts%(room%)=cts%(room%) OR 8
:RETURN
1610 IF room%=troll%(1) AND troll%(2)
=0 THEN cts%(room%)=cts%(room%) OR 4:R
ETURN
1620 RETURN
1630 REM*********inventory**********
1640 PRINT #5,"You are carrying:-"
1650 IF C%=0 THEN PRINT #5,"Nothing of
 use.":RETURN
1660 FOR N%=1 TO C%
1670 PRINT #5,"1 large ";coin$(N%);" c
oin."
1680 NEXT
1690 RETURN
1700 REM**********enter*************
1710 enter%=room%
1720 GOSUB 1550: REM troll
1730 CLS #5
1740 PEN #5,1
1750 RESTORE 2720
1760 room%=0
1770 WHILE room%<>enter%
1780 GOSUB 2190: REM find
1790 WEND
1800 crip$="You are standing "+des$:PR
INT #5:PRINT#5, crip$
1810 GOSUB 2260: REM contents
1820 last%=room%
1830 GOSUB 1850: REM directions
1840 RETURN
1850 REM*********directions***********
1860 PEN #5,2
1870 IF room%<>14 THEN PRINT #5:PRINT
#5,"I see a way ";
1880 RESTORE 3290
1890 FOR N%= 1 TO 4
1900 READ des$
1910 IF dir%(N%)<> 0 THEN PRINT #5, de
s$;",";
1920 NEXT
1930 IF dir%(N%) <>0 THEN PRINT #5, CH
R$(8);".";CHR$(8) ELSE PRINT #5, " "
1940 IF dir%(5) <>0 THEN PRINT #5:PRIN
T #5,"A set of magical steps lead upwa
rds."
1950 IF dir%(6)<> 0 THEN PRINT #5:PRIN
T #5,"A hole in the floor leads down."
1960 RETURN
1970 REM**********get*************
1980 IF room%<>5 AND cts%(room%)=0 THE
N PRINT #4,"Nothing here worth taking.
":RETURN
1990 IF (cts%(room%) AND 1)=1 THEN PRI
NT #4,"Get large coin.":PRINT #4,"Take
n":PRINT #4,"Shloop!!!.. the pedestal
vanishes.":cts%(room%)=cts%(room%) AND
 254:coin%(C%)=0:C%=C%+1:RETURN
2000 IF (cts%(room%) AND 2)=2 THEN PRI
NT #4,"Get large jewel.":PRINT #4,"Can
't the things stuck.":RETURN
2010 IF (cts%(room%) AND 4)=4 OR (cts%
(room%) AND 8)=8 THEN PRINT #4,"Get bu
rly troll?":PRINT #4,"Nice try!"
2020 IF (cts%(room%) AND 8)=8 THEN PRI
NT #4,"Get large jewel.":PRINT #4,"The
 troll won't let me!"
2030 IF room%=5 THEN PRINT #4,"Get ske
leton.":PRINT #4,"One skeleton taken."
:N%=(INT(RND*2)+1)+4:GOSUB 2390
2040 RETURN
2050 REM**********drop*************
2060 IF C%=0 THEN PRINT #4,"Drop what?
```

```
":RETURN
2070 PRINT #4,"Drop large coin."
2080 IF (cts%(room%) AND 2)=2 OR (cts%
(room%) AND 1)=1 THEN PRINT #4,"The ";
coin$(C%);" coin sticks to your finger
s.":RETURN
2090 IF (room%<>23 AND room%<>10) THEN
 coin%(C%)=room%
2100 IF room%=23 THEN coin%(C%)=14
2110 IF room%=10 THEN coin%(C%)=0
2120 PRINT #4," Dropped.":C%=C%-1
2130 IF room%=23 AND cts%(14)=0 THEN P
RINT #4,"The coin "coin$(C%+1);" rolls
 away down a hole in the floor.":cts%(
14)=1:RETURN
2140 IF room%=10 AND (cts%(10) AND 2)=
0 THEN PRINT #4,"Buuuuurp.":PRINT #4,"
Just what I needed.":RETURN
2150 IF (cts%(room%) AND 4)=4 THEN PRI
NT #4,"The burly troll shrieks and run
s away.":troll%(1)=INT(RND*27)+1
2160 IF (cts%(room%) AND 8)=8 THEN PRI
NT #4,"The burly troll walks off sulki
ng.":troll%=INT(RND*27)+1:GOSUB 1550
2170 PRINT #4,"Whoosh! as the ";coin$(
C%+1);" coin drops a pink":PRINT #4,"p
edestal rises to catch it.":cts%(room%
)=cts%(room%) XOR 1
2180 RETURN
2190 REM********find*****************
2200 READ room%
2210 FOR N%=1 TO 6
2220 READ dir%(N%)
2230 NEXT
2240 READ des$
2250 RETURN
2260 REM********contents*************
2270 RESTORE 3300
2280 N%=1:WHILE N%<>16
2290 READ des$
2300 IF (cts%(room%) AND N%)=N% THEN P
RINT #5, des$
2310 N%=N%*2
2320 WEND
2330 RETURN
2340 REM ***********quit***********
2350 PRINT #4,"QUIT!!!":PRINT #4,"Are
you really sure?"
2360 WHILE INKEY$="":WEND
2370 IF INKEY$="y" OR INKEY$="Y" THEN
CLS:PRINT #4,"Bye.":END
2380 RETURN
2390 REM ********messages***********
2400 IF N%=1 THEN PRINT #4,"Suddenly,
a massive hole opens up in":PRINT #4,"
the ground beneath your feet. The shoc
k":PRINT #4,"of landing kills you."
2410 IF N%=2 THEN PRINT #4,"A large pe
ndulum swings in from the roof";"and n
eatly cleaves your skull."
2420 IF N%=3 THEN PRINT #4,"A massive
bouler rolls in and flattens":PRINT #4
, "you."
2430 IF N%=4 THEN PRINT #4,"A large sp
ear glides down from above and";"kills
 you."
2440 IF N%=5 THEN PRINT #4,"The skelet
on didn't like being pulled":PRINT #4,
 "off the wall, and you now find youse
lf":PRINT #4,"chained there too!"
2450 IF N%=6 THEN PRINT #4,"The skelet
on promptly grabs you by the":PRINT #4
,"neck and throttles you."
2460 IF N%<=6 THEN PRINT #4:PRINT #4,"
So you lose again sucker!!":END
2470 IF N%=7 THEN PRINT #4,"Congratula
tions, you win. The elvies":PRINT #4,"
mass around you and carry you away to"
:PRINT #4,"a hero's welcome.":END
2480 RETURN
2490 REM ********initialise**********
2500 WINDOW #5,1,40,1,13
2510 WINDOW #4,1,40,16,25
2520 WINDOW #6,1,40,14,15
2530 LOCATE #6,1,1:PRINT #6,STRING$(40
,"-")
2540 C%=0:N%=0:T%=0:X%=0:Y%=0
2550 room%=13:last%=17
2560 DIM dir%(8),cts%(27),troll%(2),je
w%(3),coin%(3),coin$(3)
2570 des$= STRING$(255," ")
2580 RESTORE 3320
2590 FOR N%= 1 TO 3
2600 READ coin$(N%)
2610 NEXT
2620 jew%(1)=3:jew%(2)=10:jew%(3)=25
2630 FOR N%=1 TO 27
2640 READ cts%(N%)
2650 NEXT
2660 troll%(1) = 17:troll%(2) = 1
2670 RETURN
2680 REM *******centre text***********
*
2690 x%=(40-LEN(msg$))/2  +1
2700 LOCATE x%,y%:PEN c%:PRINT msg$
2710 RETURN
2720 DATA 1,2,0,4,0,0,0
2730 DATA "at the bottom of a deepwell
."
2740 DATA 2,3,1,0,0,0,0
2750 DATA "in a misty North/South pass
age."
2760 DATA 3,0,2,6,0,0,0
2770 DATA "at the end of a misty  pass
age."
2780 DATA 4,0,0,7,1,0,0
2790 DATA "a strange room with a  very
 high ceiling."
2800 DATA 5,0,0,8,0,0,0
2810 DATA "in the torture chamber.The
room is full of weird equipment.    A
skeleton hangs from the wall and the
place is littered with cobwebs."
2820 DATA 6,0,0,9,3,0,0
2830 DATA "in a dank East/West    pass
age. A cloud of green mist enters   fr
om the West."
2840 DATA 7,8,0,0,4,16,0
2850 DATA "in the southeast cornerpass
age."
2860 DATA 8,9,7,0,5,0,0
2870 DATA "in a small dark        ante
-chamber."
2880 DATA 9,0,8,0,6,0,0
2890 DATA "at one end of a damp   pass
age."
2900 DATA 10,11,0,0,0,0,1
2910 DATA "in the food store.     A si
gn on the wall flashes 'FEED ME'."
2920 DATA 11,12,10,0,0,0,0
2930 DATA "at the south end of thekitc
hen."
2940 DATA 12,0,11,15,0,0,0
2950 DATA "at the north end of thekitc
hen."
2960 DATA 13,14,0,16,0,22,0
2970 DATA "in the guests lounge.  A si
gn over the north door says,        'D
ON'T GO NORTH'."
2980 DATA 14,0,0,0,0,0,0
2990 DATA "in the trap room.      A si
gn on the wall reads 'I WARNED YOU!'"
3000 DATA 15,0,0,18,12,0,0
3010 DATA "in a short connecting  hall
."
3020 DATA 16,17,0,0,13,0,0
3030 DATA "at the south end of thelong
 dining room."
3040 DATA 17,18,16,0,0,0,0
3050 DATA "in the middle of the   long
 dining room."
3060 DATA 18,0,17,0,15,0,0
3070 DATA "at the north end of thelong
 dining room."
3080 DATA 19,0,0,22,0,0,0
```

```
3090 DATA " at the west end of a long
E/W hall."
3100 DATA 20,21,0,0,0,0,0
3110 DATA "at the south end of theking
's bedroom."
3120 DATA 21,0,20,24,0,0,0
3130 DATA "at the north end of theking
's bedroom."
3140 DATA 22,0,0,25,19,0,13
3150 DATA "at the east end of a  long
 East/West hall."
3160 DATA 23,0,0,26,0,0,5
3170 DATA "in the queens dressing cham
ber."
3180 DATA 24,0,0,27,21,0,0
3190 DATA "in the queens bedroom."
3200 DATA 25,26,0,0,22,0,0
3210 DATA "in Eliza's room."
3220 DATA 26,27,25,0,23,0,0
3230 DATA "at the south end of a  long
 East/West passage."
3240 DATA 27,0,26,0,24,0,0
3250 DATA "at the north end of a  long
 North/South passage."
3260 DATA N,1,S,2,E,3,W,4,U,5,D,6,@,7
3270 DATA G,8,L,9,I,10,Q,11,w,12,P,12
3280 DATA A,13,H,14,?,255
3290 DATA North,South,East,West
3300 DATA "A tall pink pedestal with a
 large coin  on it stands on the floor
.","A tall blue pedestal with a large
 jewel on it stands on the floor.","A b
urly troll is gazing at you."
3310 DATA "A burly troll carrying a la
rge jewel,  stands looking at you."
3320 DATA Bronze,Silver,Golden
3330 DATA 1,0,0,0,0,0,0,0,0
3340 DATA 0,0,0,0,0,0,0,1,0,0
3350 DATA 0,0,1,0,0,0,0,0,0
3360 DATA 30,"You're doing fine.",40,"
Why don't you make a map?",60,"not muc
h time left now",80,"We're all gonna d
ie!",95,"Ah well, there goes another l
ife."
3370 END
3380 REM**********delay***************
3390 FOR delay = 1 TO d%*100:NEXT
3400 RETURN
```

Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

RSXs at your command

ROLAND WADDILOVE introduces Resident System Extensions

AMSTRAD Basic is rather peculiar, being superb in some areas such as the unique ability to handle interrupts, yet falling down badly in others such as the graphics department, where there is a notable absence of commands.

The operating system is quite different however, having a wealth of commands and routines for the machine code programmer, all fully documented in the Complete Operating System Firmware Specification from Amsoft.

One of the most powerful features of the Amstrad is the ability to add commands to the Basic and so overcome any deficiencies. There are few restrictions. The main difference is the syntax of the new commands, which I will describe a bit later on.

Variables and absolute values can be passed so that functions are possible, but you are restricted to passing the result back to a variable. You could not print the value directly – you have to put it in a variable and print that.

These additional commands are written as machine code subroutines which can be located almost anywhere in RAM and be any length.

The names and execution addresses of the routines are stored in a table which the operating system must be informed of so that they can be recognised and used within a Basic program.

Commands such as these are known as Resident System Extensions or RSXs for short and can be recognised by the vertical bar | which must precede them.

Program I adds seven new commands to Amstrad Basic. Their syntax and function is described later. First we will look at how they are actually constructed.

All dialects of Basic have some method of calling a machine code subroutine. On the Amstrad this takes the form of a CALL statement. Not only can a machine code program be run but an optional list of parameters may be passed as well, which can be picked up and used by the routine.

The parameters are placed after the CALL statement and are separated by commas. When the code is called they are placed in a block somewhere in RAM.

Where exactly this block is located is unimportant, as the routine is entered with the A register equal to the number of parameters and the IX register pointing to the start of the block.

The parameters can be obtained using indexed addressing with the IX register, LD B, (IX+0) would load the B register with the first item in the parameter block (low byte only).

Parameters can be either integer expressions, real expressions or the address of a variable. A two byte number is placed in the block for each parameter passed. An integer number, variable or expression is passed directly, a real number, variable or expression is first converted to an integer before being placed in the block.

By placing @ before a variable its address is placed in the parameter block allowing information to be passed back into Basic variables.

CALL &8000,@a% would pass the address of a% whereas CALL &8000,a% would pass its value. If you know where a variable is stored you can then put the result of the routine back in it.

All strings must be passed in this manner. They cannot be passed directly – the value placed in the block is the address of the string descriptor. Byte 0 of the descriptor is its length and bytes 1 and 2 contain the address at which the string is stored.

For some unknown reason Basic places the values in the parameter block in reverse order. So if you CALL &8000,a,b,c,d the first value in the parameter block is d, followed by c,b and a, two bytes each, low byte/hi byte in normal Z80 fashion.

CALL &AA00 in a program is meaningless to anyone but the programmer who wrote the routine in the first place. In fact if you have several routines it is difficult to keep track of them yourself!

It would be nice if we could give the program a name and use that instead. *Screen.dump* is far more descriptive than CALL &AA00. This is exactly the facility an RSX gives us.

An RSX is simply a CALL to a machine code subroutine, but instead of having to remember its address we can use its name instead, as if it were a Basic command.

Anything that can be done with CALL can be done with an RSX command. There is no difference. All CALL XXX,... statements can be replaced with |COMMAND,...

In order to do this an external command table must be set up, the operating system must know what the routines are called and where they are located. The table is split into two, which need not reside next to each other. One half is a table of jumps to the execution addresses of the routines and the other contains their names.

The first two bytes of the RSX table contain the address of the list of names. The last letter of each name has &80 added to its Ascii code to mark it and the name list ends with a zero byte. Four bytes of workspace are also required by the operating system.

The RSX command table in Program I is constructed as follows:

```
.rsx_table
DEFW name_table ;pointer to name list
JP reset        ;jump table
JP set_clock
JP gpen
:
:
.name_table
DEF$ 'RESE'     ;last letter has &80
DEF &D4         ;added to code
DEF$ 'SETCLOC'
DEFB &CB
:
:
DEFB 0
.workspace
DEFS 4
```

```
10 REM PROGRAM I
20 REM By R.A.Waddilove
30 REM(c)Computing With The Amstrad
40 sum=0:address=&8000
50 FOR i=1 TO 36
60 READ code$
70 FOR j=1 TO 25 STEP 2
80 byte=VAL("&"+MID$(code$,j,2))
90 POKE address,byte
100 sum=sum+byte:address=address+1
110 NEXT
120 NEXT
130 IF sum<>50246 THEN PRINT"Error"
140 END
150 DATA 010980218580C3D1BC2F80C3E6
160 DATA 80C3A180C3C680C3DC80C30381
170 DATA C31581C32681000000000000000
180 DATA 00000000000000005245534504
190 DATA 534554434C4F43CB475045CE47
200 DATA 50415045D24745D4464C5553C8
210 DATA 46494CCC0000000000000000000
220 DATA 00000000000000000000000000
230 DATA 00000000000000000000000000
240 DATA 00000000000000000000000000
250 DATA 00000000000000002195807ECD5A
260 DATA BB23A720F8C952535820657272
270 DATA 6F720D0A00210000110000A7CA
280 DATA 10BD3D2009DD6E00DD6601C310
290 DATA BDDD5E00DD5601DD6E02DD6603
300 DATA C310BDFE02C28980DD7E00CDDE
310 DATA BB3E17CD5ABBDD7E02C35ABB3D
320 DATA C28980DD7E00C3E4BBCD06B9F5
330 DATA CD37BDCD00BBCDFFBBCD65BCCD
340 DATA A7BCCD5ABBCD4EBBF1C30CB93D
350 DATA C28980CD06BBDD6E00DD660177
360 DATA 233600C93DC28980DD7E00A7C2
370 DATA A7BCCD09BB38FBC9FE03C28980
380 DATA DD7E00CDDEBBDD6E02DD6603DD
390 DATA 5E04DD5605D5ED53C78122C981
400 DATA CDF0BB32CB813CCDE4BBCD11BC
410 DATA 3E0438063E0228023E0132CC81
420 DATA CD9581ED42CD788128F6D1ED53
430 DATA C781CD958109CD788128F7C922
440 DATA C781EB2AC981CDF0BB21CB81BE
450 DATA C9ED5BC781E5CDF0BB21CB81BE
460 DATA E1C92AC9812323CD878128F92B
470 DATA 2BE52AC9812B2BCD878128F923
480 DATA 23ED5BC781CDC0BBE1ED5BC781
490 DATA CDF6BB2AC781AFED4BCC8147C9
500 DATA 00000000000000000000000000
```

Program I

The operating system must be informed of the presence of any RSXs by loading the BC registers with the address of the RSX table, the HL registers with the address of the workspace and calling KL LOG EXT which is at &BCD1. The first nine bytes of the code in Program I are:

```
LD BC,rsx_table
LD HL,workspace
JP &BCD1        ;log command table &
return
```

This is located at &8000, so CALL &8000 enables the RSX command table. A word of warning though. RSX tables are linked so that more than one can be active at the same time. Be careful not to enable the same table twice.

If you do, then the second time it is linked to the first but as they both occupy the same place the operating system gets in a terrible muddle when looking for a command.

The simplest command in Program I is |GRAPER,colour which can be used to set the graphic paper colour. This is a typical example of a simple RSX command:

```
.gpaper         ;set graphic paper
colour
DEC A           ;1 parameter?
JP NZ,error     ;jump to error routine
if not
LD A,(IX+0)     ;get colour
JP &BBE4        ;call GRA SET PAPER &
return
```

|GPAPER could be entered as |gpaper – it will be changed to upper case anyway by Amstrad Basic – and must be followed by one parameter which can be anything except a string.

The RSX commands added by Program I are shown in Table I.

Program II demonstrates the use of some of these new commands.

Space has been left for you to add your own routines. If you have a look at Program I you will see quite a few zero bytes – this is where you add your own command names and jumps.

To do this place a JP to the start of the code at &8020, and put the name (upper case, with last letter+&80) at &8052. The code can be placed anywhere from &81CD onwards.

You might like to try DEEK and DOKE . . . a double byte, 16 bit PEEK and POKE for starters. Can anyone add | CIRCLE,x,y,r?

There is room for up to five extra RSX commands, so get cracking.

```
10 REM PROGRAM II
20 DEFINT a,i,r,x,y
30 x=320:y=100:r=100
40 MODE 0:INK 0,0:BORDER 0
50 PRINT:PRINT SPC(5)"POLYGONS"
60 PRINT:PRINT "  How many sides ?"
70 PRINT:PRINT SPC(5)"( 4 - 9 )"
80 |FLUSH,0:a=0
90 WHILE a<52 OR a>57:|GET,@a:WEND
100 DEG
110 MOVE x+r,y
120 |GPEN,0,1
130 FOR i=0 TO 360 STEP 360\(a-48)
140 DRAW x+r*COS(i),y+r*SIN(i)
150 NEXT
160 |FILL,x,y,1+INT(RND*15)
170 |SETCLOCK
180 WHILE TIME<900:WEND
190 RUN
```

Program II

|RESET	Resets computer but retains program.
|SETCLOCK [, low [,high]]	Sets the clock to zero if no parameters, to low where low<65536 if one parameter, to low+65536*high if two parameters.
|GPEN,option,colour	Sets the graphic pen colour and the plotting option.
|GPAPER,colour	Sets the graphic paper colour.
|GET,@integer_variable%	Waits for a key to be pressed then places its code in integer_variable%.
|FLUSH,buffer	Flushes sound buffer if parameter=1, flushes keyboard buffer if parameter=0.
|FILL,x,y,colour	Simple fill routine, origin must be at 0,0.

Table I

TRONN CYCLES

Dodge your opponent's high-powered laser net in this exciting duel to the death by ARAMELLO CHAPMAN

THE sun rises in the East as usual – but you know that today is special. Today is the day of the game.

It has been 100 years since mankind decided to give up war and as a substitute invented the game of all games – Tronn Cycles.

Following in your father's footsteps, you have decided to enter the games in search of fame and glory.

The largest crowds in the games' history are present. They have come from all corners of the world to see the Earth's most popular entertainment.

You walk into the arena to the sound of cheering and greet your opponent. As you get into your

ROUTINES

60	Main routine. Reads keyboard, checks collision and deducts fuel.
270	Prints death of both players.
310	Prints death of blue player.
360	Prints death of red player.
410	End of game routine. Waits until Space is pressed, then goes to screen choice routine.
490	Choose screen routine. Prints different screen options. Waits for keypress and goes to screen required.
630	Variable dump. Holds all important variables, fuel at start of game and players' starting positions.
710	Prints keys needed and instructions for play.
940	Sets up open plain.
980	Sets up maze of death.
1050	Sets up random maze.

VARIABLES

ti	Fuel remaining.
y1 & y2	Coordinates of blue cycle.
x1 & x2	Coordinates of red cycle.
y3 & y4	Direction in which blue cycle is moving.
x3 & x4	Direction in which red cycle is moving.
coly	Tests to make sure blue cycle has not hit anything.
colx	Tests to make sure red cycle has not hit anything.

```
10 REM******************************
20 REM********TRONN CYCLES*********
30 REM********By A.Chapman*********
40 REM(c)Computing with the Amstrad
45 LET scred=0:LET scblue=0
50 GOSUB 710:REM instructions
60 REM*******MAIN ROUTINE*******
70 PLOT y2,y1,3:PLOT x2,x1,2
80 SOUND 1,1000,5,15,0,0,1:SOUND 2,340
0,5,15,0,0,1
90 IF INKEY(53)=0 THEN LET x3=1:LET x4
=0
100 IF INKEY(62)=0 THEN LET x3=-1:LET
x4=0
110 IF INKEY(63)=0 THEN LET x4=1:LET x
3=0
120 IF INKEY(71)=0 THEN LET x4=-1:LET
x3=0
130 IF INKEY(4)=0 THEN LET y3=1:LET y4
=0
140 IF INKEY(5)=0 THEN LET y3=-1:LET y
4=0
150 IF INKEY(14)=0 THEN LET y4=1:LET y
3=0
160 IF INKEY(13)=0 THEN LET y4=-1:LET
y3=0
170 LET x1=x1+x3*2:LET y1=y1+y3*2:LET
x2=x2+x4*4:LET y2=y2+y4*4
180 LET ti=ti-1:IF ti=0 THEN GOTO 270
190 LOCATE 16,1:PEN 12:PRINT ti
200 LET colx=TEST(x2,x1)
210 LET coly=TEST(y2,y1)
220 IF colx=4 OR coly=4 THEN GOTO 240
ELSE IF colx<>0 AND coly<>0 THEN GOTO
270
230 IF colx<>0 THEN GOTO 310 ELSE IF c
oly<>0 THEN 360
240 GOTO 70
270 REM***OUT OF FUEL/TWIN CRASH***
280 REM******************************
285 MODE 1:PEN 7
290 IF ti<1 THEN PRINT"    Hard luck y
ou both ran out of fuel.":GOTO 430
295 PRINT"    Hard luck both of you ar
e dead."
300 GOTO 430
310 REM*****Death of Blue cycle*****
320 REM******************************
330 MODE 1:PEN 3:PRINT"          Well
done red cycle."
335 LET scred=scred+1
340 PRINT"        The other player R.
I.P"
350 GOTO 430
360 REM*****Death of Red cycle*****
370 REM******************************
380 MODE 1:PEN 2:PRINT"          Well d
one blue cycle."
385 LET scblue=scblue+1
390 PRINT"        The other player R.I
.P"
400 GOTO 430
410 REM**************************
420 REM******END OF GAME********
430 REM**************************
434 LOCATE 18,8:PEN 1:PRINT"SCORE":LOC
```

machines of death the crowd goes quiet as if in an act of respect for the loser.

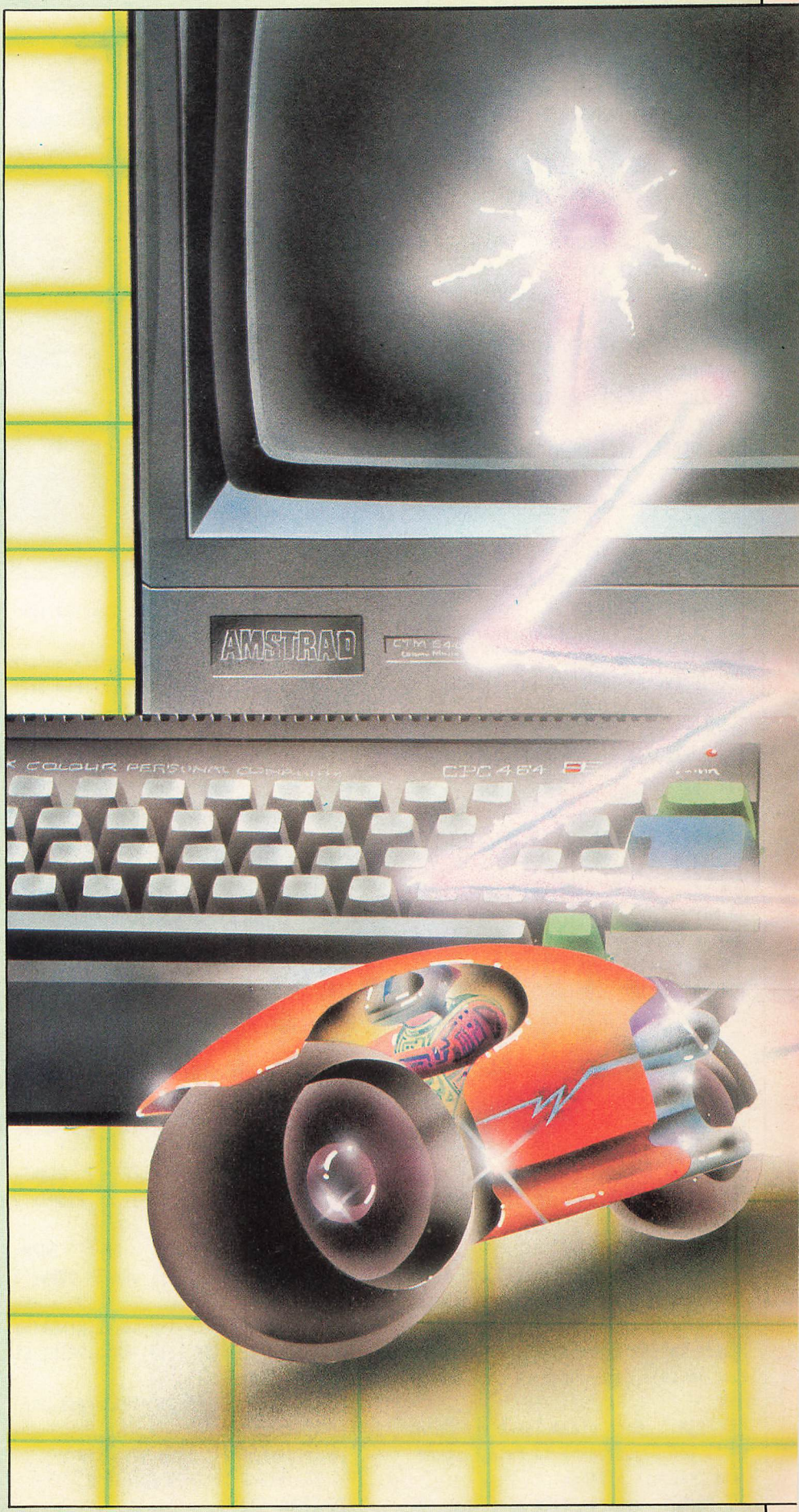
The game of Tronn Cycles is a simple one. The two opposing forces, each in their Tronn Cycles, race around the playing area. Each cycle leaves a deadly net of high-powered laser light in its wake. If either fighter crashes into it he immediately ceases to exist.

A recent development in the game is the random maze, which is said to be the most deadly of the three available, the others being open plain and maze of death.

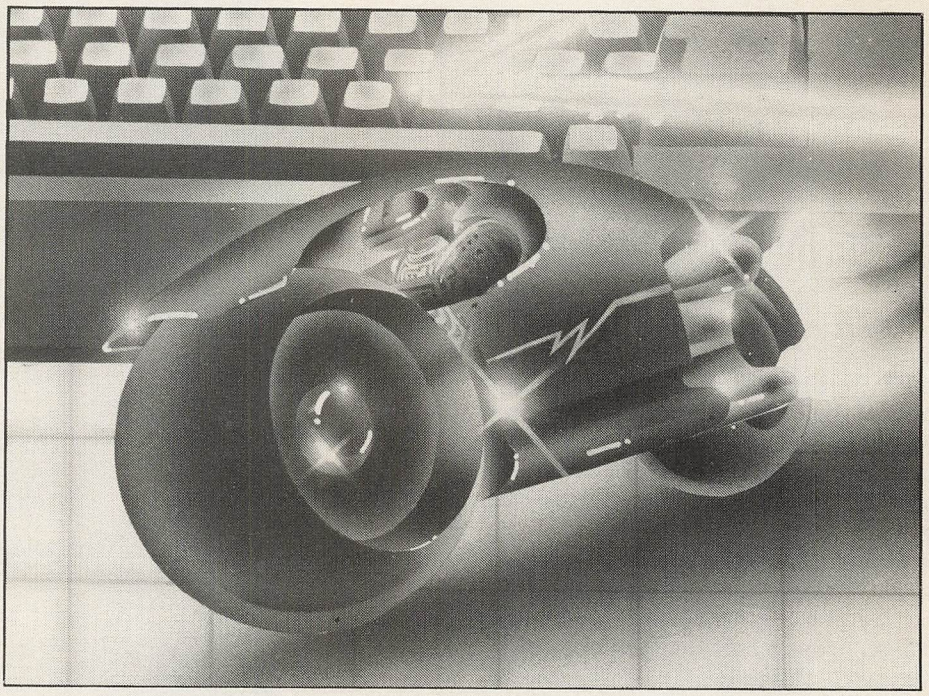
Each fighter should keep a wary eye on the fuel level because if it reaches zero both players will be exterminated. Both cycles have the same amount of fuel and travel at the same speed so they will run out at the same time.

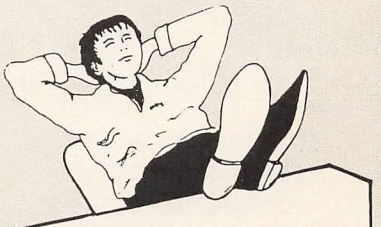
Finally, each player should not stay on the same part of the screen for too long as this gives the opponent time to trap the other Tronn Cycle.

```
ATE 18,9:PEN 3:PRINT"=====
435 LOCATE 11,11:PEN 2:PRINT"BLUE.....
......."scblue
436 LOCATE 11,13:PEN 3:PRINT"RED.....
......."scred
440 LOCATE 13,15:PEN 2:PRINT"Play agai
n(Y/N)?"
441 IF INKEY$<>"" THEN GOTO 441
442 IF INKEY$="" THEN GOTO 442
443 IF INKEY(43)=0 THEN GOTO 450
444 IF INKEY(46)=0 THEN INK 0,1:PAPER
0:PEN 1:CLS:BORDER 1:END
445 GOTO 441
450 LOCATE 13,17:PEN 1:PRINT"Change Ma
ze(Y/N)?"
451 IF INKEY$<>"" THEN GOTO 451
452 IF INKEY$="" THEN GOTO 452
453 IF INKEY(46)=0 THEN GOTO 460
454 IF INKEY(43)=0 THEN GOSUB 490:GOTO
 60
455 GOTO 451
460 GOSUB 630
```



```
465 ON maze GOSUB 940,980,1050
470 GOTO 60
490 REM*******Choose screen*******
500 INK 1,24:CLS:PEN 2:LOCATE 12,4:PRI
NT"--- ------- ------":PEN 3:LOCATE 12
,5:PRINT"THE PLAYING FIELDS"
510 LOCATE 12,6:PEN 1:PRINT"--- ------
- ------"
520 LOCATE 1,9:PRINT"    There are 3 t
ype  of battle field."
530  LOCATE 14,12:PEN 1:PRINT"1.OPEN PL
AIN"
540 LOCATE 14,14:PEN 3:PRINT"2.MAZE OF
 DEATH"
550 LOCATE 14,16:PEN 2:PRINT"3.RANDOM
MAZE"
560 LOCATE 10,20:PRINT"Enter number 1,
2 or 3."
570 LOCATE 10,21:PEN 3:PRINT STRING$(2
2,"_")
580 LET k$=INKEY$
590 IF k$="1" THEN LET maze=1:GOSUB 63
0:GOSUB 940:RETURN
600 IF k$="2" THEN LET maze=2:GOSUB 63
0:GOSUB 980:RETURN
610 IF k$="3" THEN LET maze=3:GOSUB 63
0:GOSUB 1050:RETURN
620 GOTO 580
630 REM*******VARIABLES*******
640 MODE 0
660 LET ti=999
670 LET x1=24:LET x2=319:LET y1=350:LE
T y2=319
680 LET x3=1:LET x4=0:LET y3=-1:LET y4
=0
700 RETURN
710 REM*******INSTRUCTIONS*******
720 MODE 1:INK 0,0:INK 1,24:INK 2,11:I
NK 3,6:CLS:INK 8,24:PEN 2:CLS:BORDER 3
730 PRINT:LOCATE 15,2:PEN 2:PRINT"----
--------"
740 PEN 1:LOCATE 15,3:PRINT"TRONN CYCL
ES"
750 LOCATE 15,4:PEN 3:PRINT"----------
--"
760 LOCATE 12,5:PEN 3:PRINT"By Aramell
o Chapman"
765 PEN 1:PRINT:PRINT
770 PRINT"  The year is 3075 and it is
 the 100th  Tronn Cycle championships.
As two heroic gladiators you enter the
 games in search of fame and glory.How
ever in this game there is only one wi
nner!!! "
780 PRINT"  Each player must use both
skill and  judgement to outwit,corner
 and trap his opponent."
785 PRINT"  Beware:-make sure that you
 don't fall into the same trap as you
have laid for your enemy."
790 PEN 1:PRINT TAB(16);"---------":PE
N 3:PRINT TAB(16);"GOOD-LUCK":PEN 2:PR
INT TAB(16);"---------"
800 LOCATE 1,22:PEN 3:PRINT STRING$(40
,"-"):LOCATE 9,23:PEN 2:PRINT"Enter <S
PACE> to continue.":LOCATE 1,24:PEN 1:
PRINT STRING$(40,"-")
810 IF INKEY(47)<>0 THEN GOTO 810
820 CLS:PRINT:PEN 1
840 LOCATE 16,5:PEN 3:PRINT"--------":
LOCATE 16,6:PEN 1:PRINT"CONTROLS"
845 LOCATE 1,8:PEN 3:PRINT STRING$(40,
CHR$(154))
850 LOCATE 16,7:PEN 2:PRINT"--------":
PEN 2:LOCATE 3,9:PRINT "BLUE PLAYER
 ";:PEN 3:PRINT TAB(26);"RED PLAYER"
855 LOCATE 1,15:PEN 2:PRINT STRING$(40
,CHR$(154))
860 LOCATE 3,10:PEN 1:PRINT"---- -----
-":LOCATE 26,10:PRINT"--- ------"
870 PEN 2:PRINT:PRINT "LEFT-'Z' RIGHT-
'X'";:PEN 3:PRINT"   LEFT-'1' RIGHT-'2
'"
880 PEN 2:PRINT:PRINT "   UP-'F' DOWN-
'C'";:PEN 3:PRINT"      UP-'6' DOWN-'3
'"
895 LOCATE 1,17:PEN 1:PRINT STRING$(40
,CHR$(154)):LOCATE 1,19:PEN 3:PRINT ST
RING$(40,CHR$(154))
900 LOCATE 8,18:PEN 2:PRINT"press any
key to continue."
910 IF INKEY$<>"" THEN GOTO 910
915 IF INKEY$="" THEN GOTO 915
920 GOSUB 490
930 RETURN
940 REM*****OPEN PLAIN*****
950 CLS: LOCATE 1,1:PEN 3:PRINT"FUEL R
EMAINING-"
960 PEN 12:PLOT 0,382,1:DRAW 639,382,1
:DRAW 639,0,1:DRAW 0,0,1:DRAW 0,382,1
970 RETURN
980 REM****MAZE OF DEATH****
1000 GOSUB 940
1010 LET q$=CHR$(214)+CHR$(215)
1020 LET w$=CHR$(213)+CHR$(212)
1030 FOR r=3 TO 24 STEP 4: FOR q=2 TO
19 STEP 3:PEN 12:LOCATE q,r:PRINT q$:L
OCATE q,r+1:PRINT w$:NEXT q:NEXT r
1040 RETURN
1050 REM*******RANDOM MAZE*******
1070 GOSUB 940
1080 FOR w=1 TO 100:LET p=INT(RND*20)+
3:LET q=INT(RND*20)+3:IF p=10 THEN LET
 p=3
1090 LOCATE p,q:PEN 7:PRINT CHR$(224):
NEXT w
1100 RETURN
```





Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

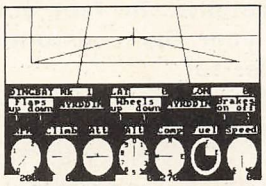
3D LANDMARKS YOU CAN FLY AROUND

SUPERB REAL TIME SIMULATION

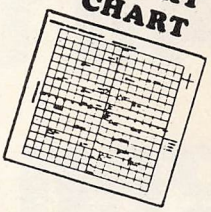
MYRDDIN FLIGHT SIMULATION

AMSTRAD CPC 464

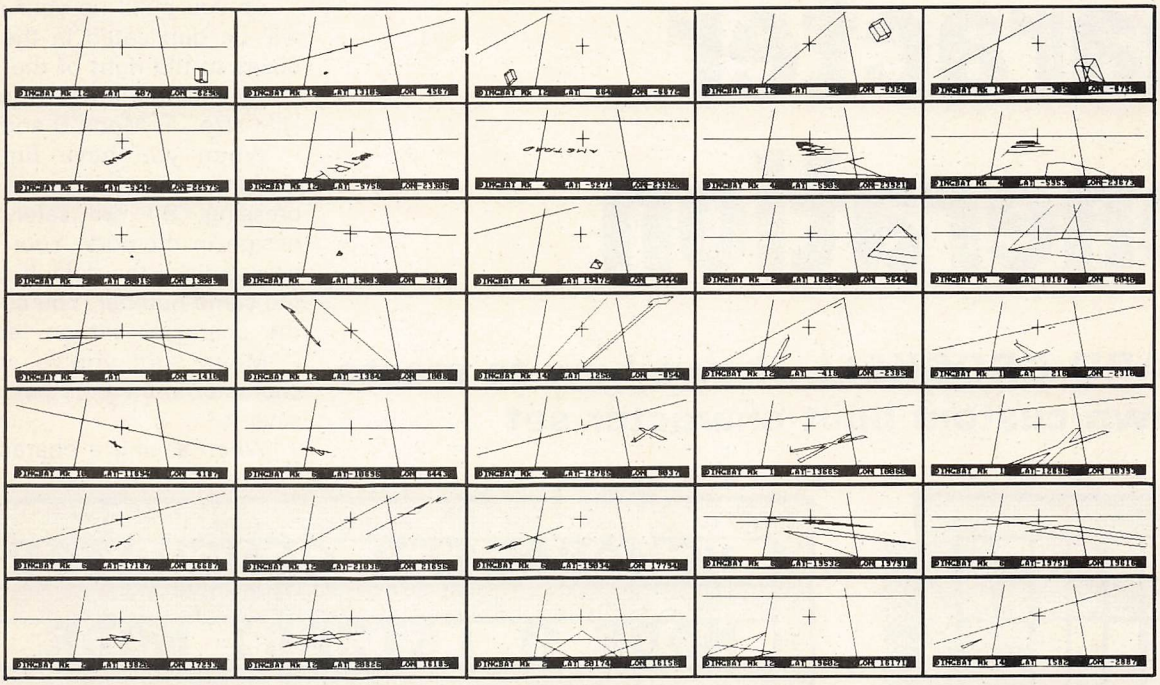
MANUAL



FULL SCREEN DISPLAY



Here are some screens from a typical flight showing the view from the cockpit (top half of screen) produced as printouts of the actual simulator.



A real time simulation with 3D graphics uses a massive 64000 x 64000 longitude & latitude flying area, making each flight completely different. Developed under pilot instruction to give realistic flight effect. The view through the cockpit gives moving 3D graphics.

Comprehensive instrument panel with moving needle meters & digital displays. 15 aircraft types with varying control sensitivities & speeds of between 100 – 500 knots.

3 runways available for refuelling, take off & landing. Ground and landmark orientation correct with all flying attitudes (rolls etc.).

The 3D graphics are still accurate when you fly upside down.

3D landmarks you can fly around.

Comes complete with manual & fully detailed chart of landmarks & airfields.

Joystick or keyboard operation.

If your local dealer doesn't have it in stock yet, order from us direct. For despatch within 48 hrs. (usually 24 hrs.).

MYRDDIN SOFTWARE, PO BOX 61, SWINDON, WILTS.
Telephone: (0793) 40661

Please send me ..... Flight Simulator(s) by return of post for the Amstrad CPC 464

Name .....

Address .....

..... Postcode .....

Cheque enclosed for £11.95 (in. P.P.)
OR Debit my Access A/C No.:-

OR Telephone through your Access Order.

Signed.....

All the gen on character generation

RICHARD CROSKELL shows how to create your own custom built character set

CHARACTER Generator is a useful utility that allows you to re-design any individual character to your own specifications. In fact you can create your own custom made character set.

Facilities are included for rotating, inverting and vertical or horizontal mirroring your characters. The data for your design can be output to the screen or printer, and saved or loaded from cassette or disc.

The program displays an 8 x 8 grid containing a flashing cursor in its top left hand corner. You design your characters by filling in the relevant cells of this grid using a joystick or the cursor keys to move the flashing cursor.

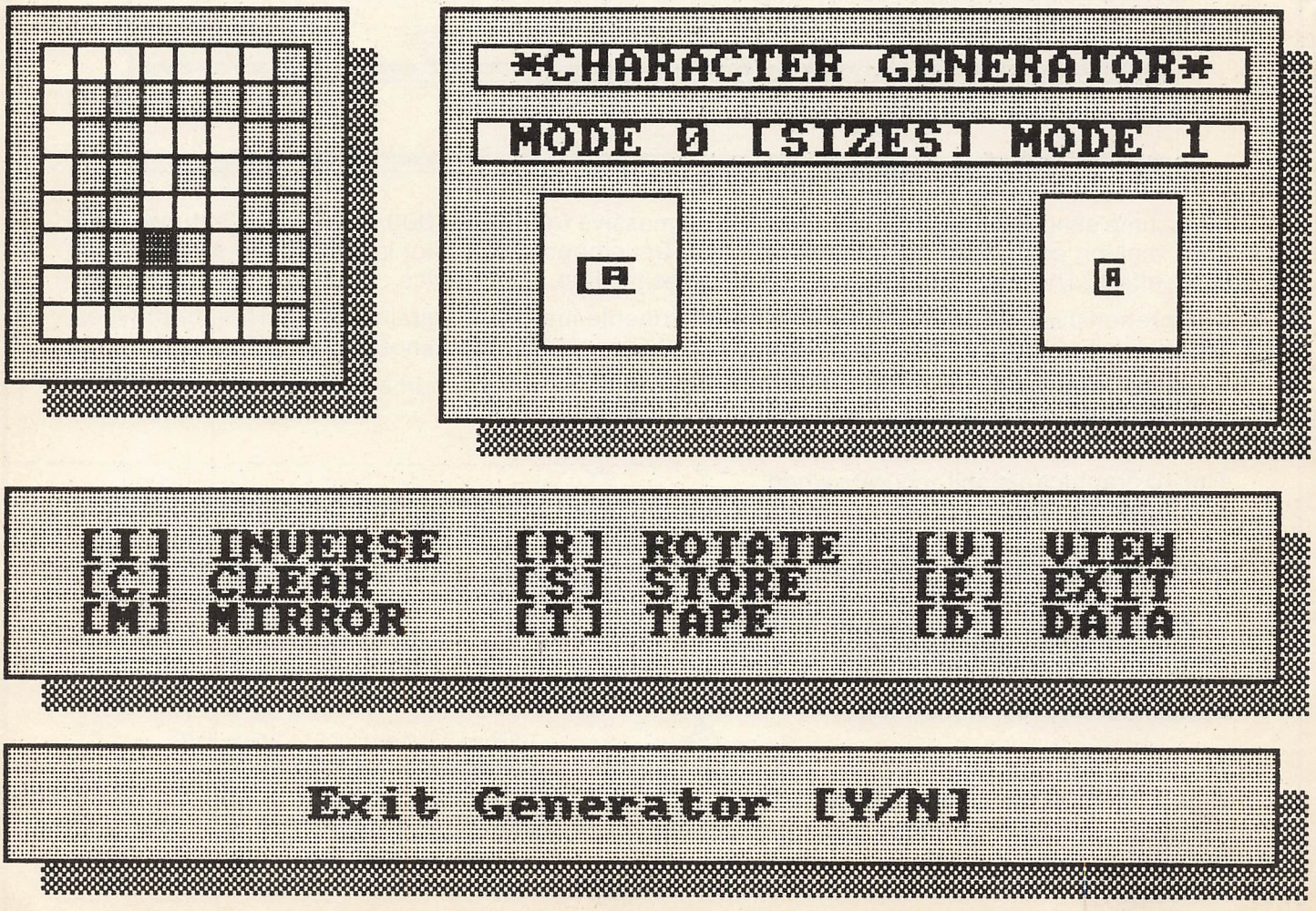
The current position can be set using the fire button or Copy key. Pressing once more will clear that cell.

As you build up your character it will be duplicated in the two small boxes to the right of the grid. These show you how the character appears "life-size" in Mode 0 and Mode 1.

When you have finished your design, using the [S]tore option (by pressing 'S') will safely store the design in memory. Your design will now replace the default character of the same number. You can also save the character set on tape or disc.

Make sure you've stored your character in memory before trying to save it.

When saving a character set the



generator stores it as a file on cassette. However this file contains more data than just the new character set. If you want to create an appropriate file for inclusion in your own program you must follow this three step procedure:

- Reset the micro using Esc/Ctrl/Shift.
- Load the program in which you wish to use the character set and make its first line:

```
10 CLS: SYMBOL AFTER 32: LOAD"!",
42240
```

then save the amended program and don't rewind the tape – we'll call this the master tape. You're going to put the data on the tape *after* this program.

- Now insert the tape that contains the file of characters saved from the generator and run a program consisting of the lines:

```
10 CLS:SYMBOL AFTER 32: LOAD"!",
42240
20 SAVE "filename",b,42240,1792
```

This will load the file into memory. When you get the prompt to press record and play, put back the un-rewound master tape in the cassette before proceeding. This will then save the character file where you want it.

The options listed below give a lot of flexibility when designing characters and are displayed continuously in the centre of the screen for quick reference. To select any option simply press the corresponding key.

[M]irror – Allows the contents of the design grid to be mirrored about its horizontal or vertical axis. Select [V] or [H] when prompted.

[R]otate – Rotates the contents of the grid by 90 degrees.

[I]nverse – Inverses the grid contents, i.e. any cell that was set will be clear and vice versa. In more simple terms it produces a negative of the original contents.

[C]lear – Clears the design from all three boxes. Don't forget to store the contents if you want to save the design, otherwise the data will be lost when this option is used.

[V]iew – Allows you to retrieve any of the characters (codes 32 – 255) from memory and place its form on the design grid for examination or modification.

[D]ata – Produces the eight character bytes in decimal form, to be directed to the [S]creen or [P]rinter when prompted. Further prompts, [C]haracter, [N]ext and [F]inish allow you to select the order in which you wish to view them.

VARIABLES

gr%	Grid square condition (1=set 0=Clear).
ro%	Condition after rotation (1=set 0=clear).
re%	Check for redefined character (1=Yes 0=No).
by%	Eight bytes that form character matrix.
gx%	Horizontal position of cursor.
gy%	Vertical position of cursor.
mx1%	Mode 0 character horizontal plot start position.
mx2%	Mode 1 character horizontal plot start position.
my1%	Mode 0 character vertical plot start position.
my2%	Mode 1 character vertical plot start position.
f	Flag for pen in design grid.
p	Flag for pen in true size grid.

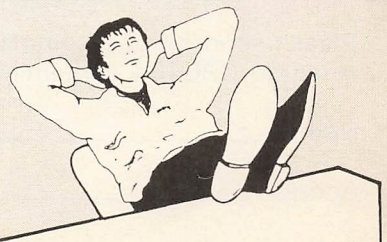
PROGRAM STRUCTURE

10-25	Title and credits.
30-50	Disable Break, select mode, colours, and set up variables.
55-90	Display flashing cursor, wait for key press.
95-150	Move flashing cursor in appropriate direction.
155-165	Cursor state (Set/clear).
170-200	Rotate option.
205-225	Invert option.
230-285	View option.
290-305	Clear option.
310-425	Cassette/Disc Save and Load.
430-480	Store option.
485-500	Exit program?
505-550	Mirror option.
555-690	Data option.
695-705	Get character matrix into array *by%*.
710-865	Poke machine code, define windows, graphics and set up screen.
870-875	Machine code data for character transfer routine.

```
10 REM ********************************
15 REM *   * CHARACTER GENERATOR *   *
20 REM *       BY R.J.CROSKELL       *
22 REM *(c)Computing with the Amstrad*
25 REM ********************************
30 MODE 1:INK 0,10:INK 1,0:INK 2,20:IN
K 3,15,1:PAPER 0:BORDER 10
35 CALL &BB48:IF PEEK(38500)=255 THEN
45 ELSE POKE 38500,255
40 SYMBOL AFTER 32:MEMORY 37999:SPEED
INK 15,15
45 DIM gr%(8,8),ro%(8,8),re%(224),by%(
8):gx%=1:gy%=1:mem%=&9470
50 f=0:p=0:mx1%=286:mx2%=534:my1%=278:
my2%=278:KEY 140,"":GOSUB 710
55 PEN #5,3:LOCATE #5,gx%,gy%:PRINT #5
,CHR$(255);
60 k$=UPPER$(INKEY$):IF k$="" THEN 60
65 IF k$=CHR$(240) OR JOY(0)=1 THEN 95
70 IF k$=CHR$(241) OR JOY(0)=2 THEN 11
0
75 IF k$=CHR$(242) OR JOY(0)=4 THEN 12
5
80 IF k$=CHR$(243) OR JOY(0)=8 THEN 14
0
```

```
85 IF k$=CHR$(224) OR JOY(0)=16 THEN 1
55
90 ON INSTR("RIVCTSEMD",k$) GOTO 170,2
05,230,290,310,430,485,505,555:GOTO 60
95 IF gy%=1 THEN PRINT CHR$(7);:GOTO 6
0 ELSE LOCATE #5,gx%,gy%
100 IF gr%(gx%,gy%)=1 THEN PEN #5,0 EL
SE PEN #5,2
105 PRINT #5,CHR$(255);:gy%=gy%-1:GOTO
 55
110 IF gy%=8 THEN PRINT CHR$(7);:GOTO
60 ELSE LOCATE #5,gx%,gy%
115 IF gr%(gx%,gy%)=1 THEN PEN #5,0 EL
SE PEN #5,2
120 PRINT #5,CHR$(255);:gy%=gy%+1:GOTO
 55
125 IF gx%=1 THEN PRINT CHR$(7);:GOTO
60 ELSE LOCATE #5,gx%,gy%
130 IF gr%(gx%,gy%)=1 THEN PEN #5,0 EL
SE PEN #5,2
135 PRINT #5,CHR$(255);:gx%=gx%-1:GOTO
 55
140 IF gx%=8 THEN PRINT CHR$(7);:GOTO
60 ELSE LOCATE #5,gx%,gy%
145 IF gr%(gx%,gy%)=1 THEN PEN #5,0 EL
SE PEN #5,2
150 PRINT #5,CHR$(255);:gx%=gx%+1:GOTO
 55
155 IF gr%(gx%,gy%)=1 THEN gr%(gx%,gy%
)=0:p=0:ELSE gr%(gx%,gy%)=1:p=1
160 PLOT mx1%+(gx%*4)-2,my1%-(gy%*2),p
:PLOT mx1%+(gx%*4),my1%-(gy%*2)
165 PLOT mx2%+(gx%*2),my2%-(gy%*2):GOT
O 60
170 FOR t%=1 TO 8:FOR g%=8 TO 1 STEP -
1:c%=ABS(g%-9)
175 ro%(t%,c%)=gr%(g%,t%):NEXT:NEXT
180 FOR t%=1 TO 8:FOR g%=1 TO 8:gr%(t%
,g%)=ro%(t%,g%)
185 IF gr%(t%,g%)=1 THEN PEN #5,0:p=1
ELSE PEN #5,2:p=0
190 LOCATE #5,t%,g%:PRINT #5,CHR$(255)
;:PLOT mx1%+(t%*4)-2,my1%-(g%*2),p
195 PLOT mx1%+(t%*4),my1%-(g%*2):PLOT
mx2%+(t%*2),my2%-(g%*2):NEXT:NEXT
200 gx%=1:gy%=1:GOTO 55
205 FOR t%=1 TO 8:FOR g%=1 TO 8:LOCATE
 #5,t%,g%
210 IF gr%(t%,g%)=1 THEN gr%(t%,g%)=0:
f=2:p=0:ELSE gr%(t%,g%)=1:f=0:p=1
215 PEN #5,f:PRINT #5,CHR$(255);:PLOT
mx1%+(t%*4)-2,my1%-(g%*2),p
220 PLOT mx1%+(t%*4),my1%-(g%*2):PLOT
mx2%+(t%*2),my2%-(g%*2)
225 NEXT:NEXT:gx%=1:gy%=1:GOTO 55
230 CLS #2:LOCATE #2,2,2:INPUT #2,"Vie
w which character [32-255]-",char$
235 IF VAL(char$)<32 OR VAL(char$)>255
 THEN 230 ELSE CLS #2
240 v%=VAL(char$):v%=(v%-32)*8:FOR t%=
0 TO 7:by%(t%+1)=PEEK(39000+v%+t%):NEX
T
245 FOR t%=1 TO 8:r%=1:FOR g%=7 TO 0 S
TEP -1
250 IF by%(t%)>=2^g% THEN 260
255 gr%(r%,t%)=0:GOTO 265
260 by%(t%)=by%(t%)-2^g%:gr%(r%,t%)=1
265 r%=r%+1:NEXT:NEXT:FOR t%=1 TO 8:FO
R g%=1 TO 8
270 IF gr%(t%,g%)=1 THEN PEN #5,0:p=1:
ELSE PEN #5,2:p=0
275 LOCATE #5,t%,g%:PRINT #5,CHR$(255)
;:PLOT mx1%+(t%*4)-2,my1%-(g%*2),p
280 PLOT mx1%+(t%*4),my1%-(g%*2):PLOT
mx2%+(t%*2),my2%-(g%*2):NEXT:NEXT
285 gx%=1:gy%=1:LOCATE #2,9,2:PRINT #2
,"** Information area **":GOTO 55
290 LOCATE #5,1,1:PEN #5,2:PRINT #5,ST
RING$(64,255);
295 FOR t%=6 TO 9:LOCATE #4,4,t%:PRINT
 #4,STRING$(4,32);
300 LOCATE #4,19,t%:PRINT #4,STRING$(4
,32);:NEXT
305 ERASE gr%:DIM gr%(8,8):gx%=1:gy%=1
:GOTO 55
310 CLS #2:PEN #2,1:LOCATE #2,2,1:PRIN
T #2,"Are you sure you have [S]tored a
way"
315 LOCATE #2,2,2:PRINT #2,"the design
, after you finished it ?"
320 LOCATE #2,10,3:PRINT #2,"Yes or No
 (Y or N)."
325 k$=UPPER$(INKEY$):IF k$="Y" OR k$=
"N" THEN 330 ELSE 325
330 ON INSTR("YN",k$) GOTO 335,425
335 CLS #2:LOCATE #2,4,2:PRINT #2,"Sav
e or Load Characters (S or L)"
340 k$=UPPER$(INKEY$):IF k$="S" OR k$=
"L" THEN 345 ELSE 340
345 IF k$="S" THEN 390
350 CLS #2:LOCATE #2,2,2:INPUT #2,"Fil
ename [8 Max]-",name$
355 IF LEN(name$)>8 THEN 350 ELSE CLS
#2:name$=UPPER$(name$)
360 LOCATE #2,8,2:PRINT #2,"Press PLAY
, then any key":WHILE INKEY$="":WEND
365 CLS #2:LOCATE #2,7,2:PRINT #2,"Loa
ding [";name$;"]."
370 LOAD"!"+name$:OPENIN"!"+name$+".CH
R":FOR t%=1 TO 224:INPUT #9,re%(t%):NE
XT:CLOSEIN
375 CLS #2:LOCATE #2,9,2:PRINT #2,"Pro
gram now in memory."
380 FOR t=1 TO 3000:NEXT:CLS #2:LOCATE
 #2,9,2
385 PRINT #2,"** Information area **":
GOTO 60
390 CLS #2:LOCATE #2,2,2:INPUT #2,"Fil
ename [8 Max]-",name$
395 IF LEN(name$)>8 THEN 390 ELSE CLS
#2:name$=UPPER$(name$)
400 CLS #2:LOCATE #2,6,2:PRINT #2,"Pre
ss REC+PLAY, then any key"
405 WHILE INKEY$="":WEND:CLS #2:LOCATE
 #2,7,2:PRINT #2,"Saving [";name$;"]."
410 SAVE"!"+name$,b,39000,1792:OPENOUT
"!"+name$+".CHR":FOR t%=1 TO 224:WRITE
 #9,re%(t%):NEXT
415 CLOSEOUT:CLS #2:LOCATE #2,12,2:PRI
NT #2,"Saving complete"
420 FOR t=1 TO 3000:NEXT
425 CLS #2:LOCATE #2,9,2:PRINT #2,"**
Information area **":GOTO 60
430 CLS #2:LOCATE #2,4,2:INPUT #2,"Ent
er character code [32-255]-",char$
435 IF VAL(char$)<32 OR VAL(char$)>255
 THEN 430 ELSE CLS #2
440 v%=VAL(char$):IF re%(v%-31)=0 THEN
 465 ELSE CLS #2
445 LOCATE #2,2,2:PRINT #2,"Already be
en redefined, Change [Y/N]"
450 k$=UPPER$(INKEY$):IF k$="Y" OR k$=
"N" THEN 455 ELSE 450
455 IF k$="Y" THEN 465 ELSE CLS #2:LOC
ATE #2,9,2
460 PRINT #2,"** Information area **":
GOTO 60
465 GOSUB 695:v%=(v%-32)*8:FOR t%=0 TO
 7
470 POKE 39000+v%+t%,by%(t%+1):NEXT:CL
S#2:LOCATE #2,4,2
475 PRINT #2,"Character has been store
d away.":re%(v%/8+1)=1:FOR t=1 TO 3000
480 NEXT:CLS #2:LOCATE #2,9,2:PRINT #2
,"** Information area **":GOTO 60
485 CLS #2:PEN #2,1:LOCATE #2,10,2:PRI
NT #2,"Exit Generator [Y/N]""
490 k$=UPPER$(INKEY$):IF k$="Y" OR k$=
"N" THEN 495 ELSE 490
495 IF k$="Y" THEN MODE 1:PRINT"Progra
m aborted":END
500 CLS #2:LOCATE #2,9,2:PRINT #2,"**
Information area **":GOTO 60
505 CLS #2:LOCATE #2,3,2:PRINT #2,"Ver
tical or Horizontal Plane (V/H)"
```

```
510 k$=UPPER$(INKEY$):IF k$="V" OR k$=
"H" THEN 515 ELSE 510
515 IF k$="V" THEN 525 ELSE FOR t%=8 T
O 1 STEP -1:FOR g%=1 TO 8
520 ro%(g%,t%)=gr%(g%,ABS(t%-9)):NEXT:
NEXT:GOTO 530
525 FOR t%=8 TO 1 STEP -1:FOR g%=1 TO
8:ro%(t%,g%)=gr%(ABS(t%-9),g%):NEXT:NE
XT
530 FOR t%=1 TO 8:FOR g%=1 TO 8:gr%(t%
,g%)=ro%(t%,g%):LOCATE #5,t%,g%
535 IF gr%(t%,g%)=1 THEN PEN #5,0:p=1
ELSE PEN #5,2:p=0
540 PRINT #5,CHR$(255);:PLOT mx1%+(t%*
4)-2,my1%-(g%*2),p
545 PLOT mx1%+(t%*4),my1%-(g%*2):PLOT
mx2%+(t%*2),my2%-(g%*2):NEXT:NEXT
550 CLS #2:LOCATE #2,9,2:PRINT #2,"**
Information area **":GOTO 55
555 code%=32:CLS #2:LOCATE #2,7,2:PRIN
T #2,"To Screen or Printer (S/P)"
560 k$=UPPER$(INKEY$):IF k$="S" OR k$=
"P" THEN 565 ELSE 560
565 IF k$="P" THEN 615:'ELSE CLS #2:FO
R t%=0 TO 223:IF re%(t%+1)=0 THEN 535
570 CLS #2:FOR t%=(code%-32) TO 223:IF
 re%(t%+1)=0 THEN 585
575 CLS #2:LOCATE #2,1,2:PEN #2,0:PRIN
T #2,STR$(t%+32);:PEN #2,1:FOR g%=0 TO
 7
580 a%=(t%*8):p%=PEEK(39000+a%+g%):PRI
NT #2,STR$(p%);:NEXT:GOTO 590
585 CLS #2:LOCATE #2,13,2:PEN #2,1:PRI
NT #2,STR$(t%+32);"-UNCHANGED"
590 LOCATE #2,2,3:PEN #2,1:PRINT #2,"[
F] Finished [N] Next [C] Character"
595 k$=UPPER$(INKEY$):IF k$="F" OR k$=
"N" OR k$="C" THEN 600 ELSE 595
600 ON INSTR("FNC",k$) GOTO 675,605,61
0
605 NEXT:GOTO 675
610 GOSUB 680:GOTO 570
615 CLS #2:PEN #2,1:LOCATE #2,3,2:PRIN
T #2,"Put printer on line, press any k
ey"
620 WHILE INKEY$="":WEND:'FOR t%=0 TO
223:IF re%(t%+1)=0 THEN 580
625 FOR t%=(code%-32) TO 223:IF re%(t%
+1)=0 THEN 645
630 CLS #2:LOCATE #2,1,2:PEN #2,0:PRIN
T #2,STR$(t%+32);:PEN #2,1
635 PRINT #8,"SYMBOL";STR$(t%+32);:FOR
 g%=0 TO 7:a%=t%*8:p%=PEEK(39000+a%+g%
)
640 PRINT #2,STR$(p%);:PRINT #8,STR$(p
%);:NEXT:PRINT #8,CHR$(13):GOTO 650
645 CLS #2:LOCATE #2,13,2:PEN #2,1:PRI
NT #2,STR$(t%+32);"-UNCHANGED"
650 LOCATE #2,2,3:PEN #2,1:PRINT #2,"[
F] Finished [N] Next [C] Character"
655 k$=UPPER$(INKEY$):IF k$="F" OR k$=
"N" OR k$="C" THEN 660 ELSE 655
660 ON INSTR("FNC",k$) GOTO 675,665,67
0
665 NEXT:GOTO 675
670 GOSUB 680:GOTO 625
675 CLS #2:LOCATE #2,9,2:PRINT #2,"**
Information area **":GOTO 55
680 LOCATE #2,1,2:PRINT #2,CHR$(18);:L
OCATE #2,2,2:PEN #2,1
685 INPUT #2,"Enter start code (32 to
255) ",code%
690 IF code%<32 OR code%>255 THEN 680
ELSE RETURN
695 FOR t%=1 TO 8:q%=0:z%=0:FOR g%=8 T
O 1 STEP -1
700 IF gr%(g%,t%)<>1 THEN 705 ELSE z%=
z%+2^q%
705 q%=q%+1:NEXT:by%(t%)=z%:NEXT:RETUR
N
710 RESTORE 870:FOR t%=0 TO 18:READ d:
POKE mem%+t%,d:NEXT:CALL mem%
715 SYMBOL 255,254,254,254,254,254,254
,254,0
720 WINDOW #1,2,39,15,19:PAPER #1,2:CL
S #1
725 WINDOW #2,2,39,22,24:PAPER #2,2:CL
S #2
730 WINDOW #3,2,11,2,11:PAPER #3,2:CLS
 #3
735 WINDOW #4,15,39,2,12:PAPER #4,2:CL
S #4
740 WINDOW #5,3,10,3,10:PAPER #5,2:CLS
 #5
745 PEN 1:LOCATE 3,20:PRINT STRING$(38
,207):LOCATE 3,25:PRINT STRING$(38,207
);
750 LOCATE 3,12:PRINT STRING$(10,207):
LOCATE 16,13:PRINT STRING$(25,207)
755 FOR t%=16 TO 19:LOCATE 40,t%:PRINT
 CHR$(207):NEXT
760 FOR t%=3 TO 11:LOCATE 12,t%:PRINT
CHR$(207):NEXT
765 FOR t%=3 TO 12:LOCATE 40,t%:PRINT
CHR$(207):NEXT
770 FOR t%=23 TO 24:LOCATE 40,t%:PRINT
 CHR$(207):NEXT
775 FOR t%=31 TO 159 STEP 16:PLOT t%,3
68,1:DRAWR 0,-128:NEXT
780 FOR t%=368 TO 240 STEP -16:PLOT 31
,t%:DRAWR 128,0:NEXT
785 PLOT 15,15:DRAWR 610,0:DRAWR 0,50:
DRAWR -610,0:DRAWR 0,-50
790 PLOT 15,384:DRAWR 162,0:DRAWR 0,-1
62:DRAWR -162,0:DRAWR 0,162
795 PLOT 15,94:DRAWR 610,0:DRAWR 0,82:
DRAWR -610,0:DRAWR 0,-82
800 PLOT 624,384:DRAWR -402,0:DRAWR 0,
-178:DRAWR 402,0:DRAWR 0,178
805 PLOT 238,368:DRAWR 370,0:DRAWR 0,-
18:DRAWR -370,0:DRAWR 0,18
810 PLOT 238,336:DRAWR 370,0:DRAWR 0,-
18:DRAWR -370,0:DRAWR 0,18
815 PLOT 270,304:DRAWR 66,0:DRAWR 0,-6
6:DRAWR -66,0:DRAWR 0,66
820 PLOT 510,304:DRAWR 66,0:DRAWR 0,-6
6:DRAWR -66,0:DRAWR 0,66
825 LOCATE #5,1,1:PRINT #5,CHR$(22)+CH
R$(1)
830 PEN #4,1:PAPER #4,0:LOCATE #4,2,2:
PRINT #4," *CHARACTER GENERATOR* "
835 LOCATE #4,2,4:PRINT #4," MODE 0 [S
IZES] MODE 1 "
840 FOR t%=6 TO 9:LOCATE #4,4,t%:PRINT
 #4,"    "
845 LOCATE #4,19,t%:PRINT #4,"    ":NE
XT
850 PEN #2,1:LOCATE #2,9,2:PRINT #2,"*
* Information area **"
855 PEN #1,1:LOCATE #1,3,2:PRINT #1,"[
I] INVERSE  [R] ROTATE  [V] VIEW"
860 LOCATE #1,3,3:PRINT #1,"[C] CLEAR
   [S] STORE   [E] EXIT"
865 LOCATE #1,3,4:PRINT #1,"[M] MIRROR
   [T] TAPE    [D] DATA":RETURN
870 DATA &01,&00,&07,&21,&00,&A5,&11,&
58,&98,&7E
875 DATA &12,&0B,&23,&13,&78,&B1,&20,&
F7,&C9
```



Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

You're never too young to play a Magical Adventure on the Amstrad CPC464 . . .

Based on the style of the classic computer adventures – but written so that even small children can learn to find their way around, encouraged by colourful graphics and exciting sound effects.

The pack contains a 48-page full colour storybook
PLUS
a full length multi-location adventure on cassette for only

£8.95! post free

Read the book – then play the game!

Makes an ideal present

Please send me the complete Magic Sword pack for the Amstrad CPC464 containing storybook and cassette to:

Name _____

Address _____

_____

_____

☐ I enclose my cheque for £8.95 payable to Database Publications
☐ Or debit my Access/Visa card:

No. _____

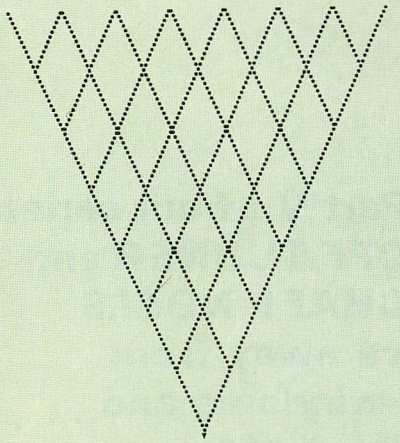
Signed _____

SEND TO: **Adventure offer, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY**

CA5

Trees

Analysed by Trevor Roberts



THIS month we use our Amstrad to draw trees. Starting with one point, or node, at the bottom of the screen we draw lines to the left and right.

The ends of each of these lines also split into two and so on. At each split, the changes in the x and y coordinates are constant, resulting in a regular pattern.

It's a simple idea, but not particularly easy to program. Trees shows one method. Can you make it more efficient?

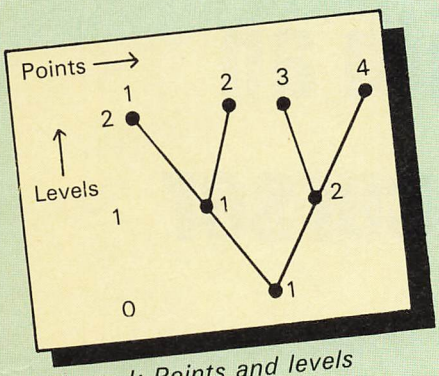


Figure I: Points and levels

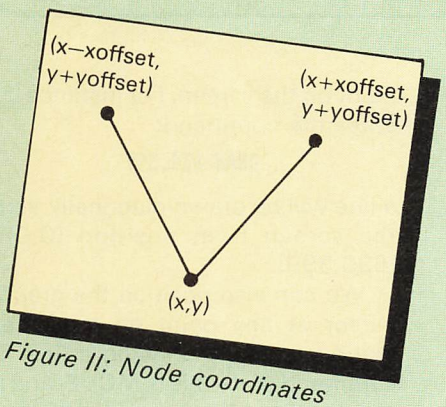
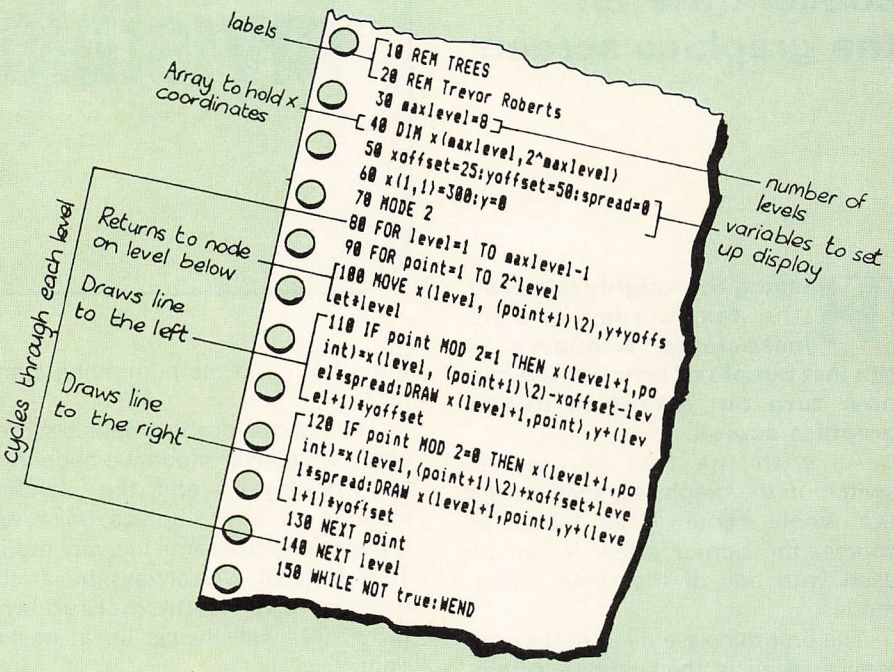


Figure II: Node coordinates



10,20 REMS that identify the program.

30 The variable *maxlevel* selects the number of levels (each line splits at a level). Have a look at Figure I to see how the levels and the number of points on them are related.

40 Horribly, this sets up a two dimensional array to hold all the possible x coordinates of the nodes. This is terribly wasteful as not all the levels have the maximum number of points. This array makes room for lots of non-existent points. Can you think of a better way of doing things?

50 Holds values for the changes in x and y coordinates for each succeeding level. *spread* just allows a little bit of variance in the way the lines split. Try giving it non-zero values and see what happens. Also vary *xoffset* and *yoffset.* Why not have them changing randomly?

60 Sets the coordinates for the initial point on level 0.

80-140 Form a FOR . . . NEXT loop which cycles once for each of the levels. Each time the coordinates for all the nodes on that level will be calculated.

90-130 Make up a second loop which does the job of working out all the coordinates of the nodes of a level and draws the lines joining them. If you look at Figure I you should see that there are 2 to the power of *level* points on each level.

100 Ensures that the graphics cursor returns to the appropriate node. Leave it out and see the result.

110 Draws the line to the left from the node. Looking at Figure I you'll see that all the odd numbered points are on the left hand of a split. The MOD checks this and draws a line to it from the node below.

120 Deals with the right splits. Figure II shows how the coordinates relate.

150 An endless loop that keeps the ready message at bay.

Plotting imaginary points with an invisible cursor

In Part V of our series GEOFF TURNER and MICHAEL NOELS move away from the windows and concentrate on the graphics screen

Having thoroughly explored the text screen and its associated windows in the last couple of articles, we can now turn our attention to the graphics screen.

As with the text screen, at switch-on the graphics screen covers the whole screen area (save, of course, the border area). It can be used with any of the three screen modes.

The first thing we need to become familiar with is the graphics screen coordinate system, as shown in Figure I. We've already met one system of coordinates, those of the text screen. However these are designed primarily to cope with the positioning of characters on the text screen, a task the system handles admirably.

We need more precision when specifying points on the graphics screen. So for graphics purposes we divide it into a 640 by 400 grid. We specify a point by giving its horizontal position first, followed by its vertical position – much as you learned at school when doing graphs.

Note that the graphics origin is situated at the bottom left hand corner of the screen and is identified as (0,0). The X axis (horizontal) ranges from 0 to 639 and the Y axis (vertical) ranges from 0 to 399.

This is completely different from how things work on the text screen, where the origin is at the top left of the screen and the numbering starts at 1.

The figures for the graphics screen apply whichever mode we happen to be working in, and the physical location of any graphics point will therefore be the same in every mode. That is (0,0) will always be at the bottom left of the screen and (639,399) will always be at its top right.

Let's try drawing some lines on the screen. To do this we imagine something called the graphics cursor, which is invisible yet occupies a specific point on the screen.

Initially when we switch on or reset the computer the graphics cursor will always reside at the origin point (0,0). We can use the DRAW command to draw a line from the cursor's present position to any other point on the screen. For instance if we enter the command:

```
DRAW 639,399
```

a line will be drawn diagonally across the screen from position (0,0) to (639,399).

We can also position the graphics cursor at any point on the screen, without actually drawing a line. The command to do this is MOVE, and it is used with the same coordinate system as the DRAW command. So we may either draw from the cursor's current position to any other, or alternatively we can simply move the cursor between the two points.

MOVE is used just like the DRAW command, by specifying X and Y coordinates. So the command:

```
MOVE 639,399
```

will move the cursor across the

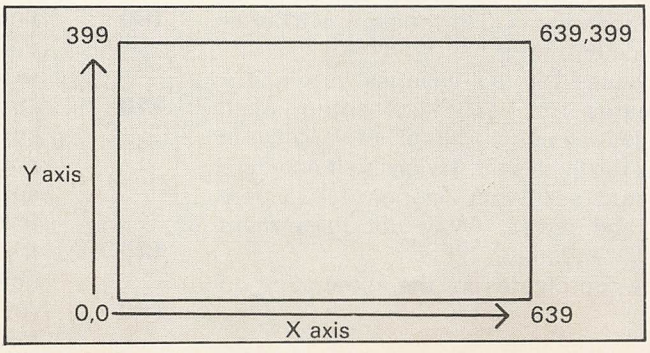


Figure I: Graphics screen coordinates

screen to the top right hand corner, but this time no line will be drawn.

Time to put theory into practice with some simple programs. Program I draws our diagonal line again, followed by a second line down the right hand edge of the screen, and finally a third line back to the starting position to form a triangle.

By the way, we have included in this program a dummy last line to prevent the text cursor and "Ready" message reappearing on the screen after the drawing is completed. This helps to keep a tidy graphics screen for the purpose of these demonstration programs. Of course you will need to press Escape twice to exit the program.

Now to illustrate how the MOVE command is used, change line 40 to:

```
40 MOVE 639,0
```

When you now run the program you will see that the triangle is incomplete, as we have only moved the cursor down the right hand side of the triangle, instead of drawing a line.

Let's try a more complex shape now. Program II forms a rectangle, with intersecting corner to corner diagonals as shown in Figure II. See how we have used a combination of MOVE and DRAW commands to produce this shape.

Of course this shape cannot be produced using DRAW commands alone (without drawing over some lines twice), so we need to throw in the occasional MOVE command to complete the shape.

By way of an exercise, see if you can reproduce the envelope shape shown in Figure III using only DRAW commands and no MOVE commands. It's something that we have all tried to do at some time with pen and paper, but now we can attempt it with a computer. We promise you it can be done!

Programs I and II have used Mode 1 to draw the shapes, but we can also use the other two Modes, 0 and 2. You'll remember that when we were dealing with the text screen, the mode controlled the number of characters we could print across the screen. The mode command also has an effect on the graphics screen, but in a slightly different way. Change line 20 in Program II to:

```
20 MODE 0
```

and run the program again. As you will see, the shape is drawn in exactly the same way, and it occupies the same position and area on the screen.

You may have noticed however that the drawn lines appear slightly thicker than when we used Mode 1. The diagonal lines appear to have a stepped effect instead of appearing as a smoother straight line.

This is because Mode 0 uses lower resolution graphics, which appear coarser than the medium resolution graphics of Mode 1. When we use a higher resolution screen mode the individual points are much smaller and the lines appear thinner.

If we want to use really high resolution graphics then we can select Mode 2. Try using Mode 2 in Program II and see the effect of using the high resolution screen.

Mode 2 allows us to create drawings with much finer detail, but as you might expect we have to pay the price for this in terms of colour.

Remember how the mode restricted our use of colour on the text screen? Unfortunately the same rules apply to using colour on the graphics screen.

The high resolution graphics of Mode 2 are only available in two colours. Therefore if we consider one colour for the background we can only really draw lines in one other colour. Of course we can select various colours from the 27 available, just like we did with the INK command in previous chapters.

Mode 0 can be used to produce multicoloured drawings in 16 colours, but we lose the fine detail of the higher resolution screens.

So it's really a case of choosing the best mode for your particular application.

We measure screen resolution in pixels, each being a separate dot of

```
10 REM PROGRAM I
20 MODE 1
30 DRAW 639,399
40 DRAW 639,0
50 DRAW 0,0
60 GOTO 60
```

Program I

```
10 REM PROGRAM II
20 MODE 1
30 DRAW 639,399
40 DRAW 639,0
50 DRAW 0,0
60 DRAW 0,399
70 DRAW 639,0
80 MOVE 0,399
90 DRAW 639,399
100 GOTO 100
```

Program II

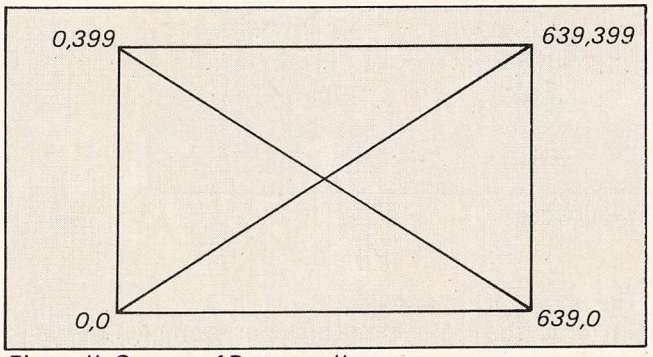


Figure II: Output of Program II

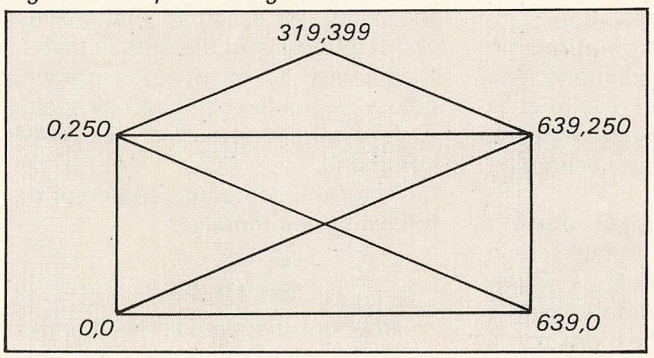


Figure III: Output of Program IV

MODE NUMBER	HORIZONTAL PIXELS	VERTICAL PIXELS
0	160	200
1	320	200
2	640	200

Table I: Mode – pixel relationship

colour on the screen. The more of these dots you can cram onto the screen the finer the lines we can draw, the better the resolution.

Table I shows the relationship between the screen resolutions. You can see that the vertical resolution is always 200 points or pixels, while the horizontal resolution varies according to the mode.

Notice that one pixel doesn't necessarily correspond to one screen coordinate point. Sometimes neighbouring screen coordinate points have to share a pixel. The more points sharing a particular pixel the less the resolution will be.

However the important point to remember is that the coordinates used in DRAW and MOVE commands are always 0 to 399 vertically, and 0 to 639 horizontally, regardless of the mode selected. This may seem a little confusing at first, but you will soon find that it works out much easier just using one set of coordinates instead of the three sets shown in Table I.

Just to make sure that you have grasped the effects of different modes, enter Program III. This steps through each mode and draws a line to the centre of the screen. Points to note are that the same coordinates are used for each mode, and that the thickness of the line varies with each mode. Line 60 simply introduces a short delay between each change.

So far all our lines have been drawn in bright yellow on a blue background. These are the normal default colours whenever we switch on the computer. Let's try introducing a change of colour into our line drawing.

```
10 REM PROGRAM III
20 FOR resolution = 0 TO 2
30 MODE resolution
40 PRINT"MODE ";resolution;
50 DRAW 320,200
60 FOR delay = 1 TO 2000:NEXT
70 NEXT resolution
80 GOTO 20
```

Program III

In the previous examples we didn't actually specify any colours, and so the computer assumed that we just wanted to draw in the default colour. If we wish to change the drawing colour we tell the computer by adding a third parameter after the DRAW command.

Reset the computer, and enter the following commands:

```
CLG
DRAW 639,399,2
```

Now our diagonal line will appear in bright cyan. You see, the third number after the DRAW command tells the computer which graphics pen to use. After issuing this command any further lines will still be drawn in bright cyan until we specify a new pen with the DRAW command.

It's very similar to selecting a text pen. Once selected, it remains in use until a further PEN command is used. You'll see if you enter:

```
DRAW 639,0
```

the second line is still drawn in bright cyan, even though we didn't specify a colour.

Now enter the command:

```
DRAW 0,0,1
```

and the colour will be restored to bright yellow to complete the triangle.

Program IV draws that envelope shape that you may have been struggling with earlier. This time we're using Mode 0 so that we can introduce a good selection of colour into the proceedings. You will see that each line is drawn in a different colour because we have appended a colour parameter to the end of each DRAW command.

```
10 REM PROGRAM IV
20 MODE 0
30 DRAW 639,0,2
40 DRAW 639,250,3
50 DRAW 0,0,4
60 DRAW 0,250,5
70 DRAW 639,250,6
80 DRAW 319,399,7
90 DRAW 0,250,8
100 DRAW 639,0,9
110 GOTO 110
```

Program IV

Try omitting some of the colour parameters in this program to see how the overall colour of the drawing is affected.

While we are on the subject of colour, let's consider how we can change the colour of the background for our graphics screen.

The CLG command is used to clear the graphics screen. It works in much the same way as the CLS command, which clears the text screen.

```
10 REM PROGRAM V
20 MODE 0
30 FOR colour=0 TO 15
40 CLG colour
50 PRINT"CLG ";colour
60 FOR delay=1 TO 1000:NEXT
70 NEXT
```

Program V

However, if you try CLG in direct mode (that is, just type it into the Amstrad and press Enter) you'll see that the Ready prompt appears on a line corresponding to the last position of the text cursor – not at the top of the screen. We may also append a pen number to this command in order to change the background colour.

Reset the computer, and enter:

```
CLG 3
```

Now as the graphics screen is cleared it changes colour to red, the pen number specified. Other colours can be selected as long as they are within the range allowed by the particular mode in use. This principle is very similar to changing the paper colour on the text screen, but of course the entire area of the graphics screen must be changed at once, as you have to clear it to change colour.

Program V demonstrates how the background colour may be changed with the CLG command. Note that the text paper colour is unaffected by this command, and that the text cursor is not returned to its home position as it is when a CLS command is issued.

You might like to try inserting some different background colours in some of the other programs. A word of warning though. Take care to avoid

drawing lines over a background colour with the same foreground colour. The lines will still be drawn, but you just won't see them!

Some very interesting effects may be obtained by introducing random elements into our drawings. We could select random colours and also draw or move to random coordinates.

In Program VI we are drawing lines in random colours to random locations. After each line is drawn we

```
10 REM PROGRAM VI
20 MODE 0
30 WHILE INKEY$=""
40 colour=INT(15*RND(1))+1
50 x=INT(639*RND(1))+1
60 y=INT(399*RND(1))+1
70 MOVE 0,0
80 DRAW x,y,colour
90 WEND
```

Program VI

move the cursor back to the origin point (0,0) and the end result is a series of lines radiating from the bottom left hand corner.

Suppose we now leave out line 80 so that the cursor is not moved back to the origin. What effect do you think this will have? Try it and see.

Alternatively we could move the cursor to another position instead of the origin. Try using these various options at line 80, and run the program again:

```
80 MOVE 320,200
80 MOVE x,0
80 MOVE x,y
80 MOVE 0,y
80 MOVE y,y
80 MOVE x,x
```

We can also produce some interesting patterns by leaving out the random functions and taking a more logical approach. Program VII again makes use of the DRAW and MOVE commands, but this time our lines are drawn in a more orderly fashion.

We start off with two fixed points (*x,y*) and (*a,b*) at extreme corners of the screen and draw a line between them. Then the values of the four variables used are changed in steps of 8 units before the next line is drawn.

There are many variations on the same theme. Program VIII is another example. Perhaps you may like to try writing some of your own routines similar to these. Try and predict what the finished pattern will look like before you run each program.

There are many effects to be obtained by creative use of the MOVE and DRAW commands. To use the computer cliche, you are limited only by your own imagination.

You should by now be familiar with the graphics screen coordinates as used with the MOVE and DRAW commands. There are, however, two related commands which use a slight variation on the graphics screen coordinate system.

These two new commands are MOVER and DRAWR. Notice that we have simply added an R to the end of MOVE and DRAW. The R stands for RELATIVE, and it simply means that the moving or drawing should be relative to the present position of the graphics cursor.

```
10 REM PROGRAM VII
20 MODE 0
30 FOR colour=1 TO 15
40 x=0
50 y=0
60 a=639
70 b=399
80 WHILE y<400 AND x<640
90 MOVE x,y
100 DRAW a,b,colour
110 x=x+8
120 y=y+8
130 a=a-8
140 b=b-8
150 WEND
160 NEXT colour
```

Program VII

Let's start off with a clear graphics screen, then we know that the cursor will be situated at (0,0). If we now issue the commands:

```
DRAW 200,200
DRAW 400,0
```

the first line will be drawn to a position 200 points across the screen and 200 points up from the base line. The second line is drawn from the end of the first line, to a position 400 points across the screen, and to vertical point 0 – that is, back to the base line. We end up with a sort of inverted V shape.

Notice that all the coordinates used in the above commands are with reference to the origin point (0,0).

```
10 REM PROGRAM VIII
20 MODE 0
30 x=0:y=200
40 a=639:b=200
50 WHILE a>320
60 MOVE x,y
70 DRAW a,b,2
80 a=a-8:b=b-4
90 WEND
100 WHILE x<320
110 MOVE x,y
120 DRAW a,b,3
130 x=x+8:y=y+4
140 WEND
150 WHILE a>0
160 MOVE x,y
170 DRAW a,b,4
180 a=a-8:b=b+4
190 WEND
200 WHILE x<639
210 MOVE x,y
220 DRAW a,b,5
230 x=x+8:y=y-4
240 WEND
250 WHILE a<320
260 MOVE x,y
270 DRAW a,b,6
280 a=a+8:b=b+4
290 WEND
300 WHILE x>320
310 MOVE x,y
320 DRAW a,b,7
330 x=x-8:y=y-4
340 WEND
350 WHILE a<640
360 MOVE x,y
370 DRAW a,b,8
380 a=a+8:b=b-4
390 WEND
400 WHILE x>0
410 MOVE x,y
420 DRAW a,b,9
430 x=x-8:y=y+4
440 WEND
450 GOTO 30
```

Program VIII

They are known as ABSOLUTE coordinates.

Let's try the same thing again using our two new commands. First of all enter a CLG command to restore the cursor back to the origin, then enter these commands:

```
DRAWR 200,200
DRAWR 400,0
```

This time the lines have been ▶

drawn relative to the graphic cursor's last position. The first line is, in fact, identical to the one in the previous example, because the cursor was initially situated at (0,0) – our CLG saw to that.

However the second line is completely different. It's been drawn horizontally from the end of the first line and ended up at a point with absolute coordinates (600,200).

What has happened is that the second line has been drawn *relative* to the last position of the cursor instead of the origin point. In fact the two sets of coordinates have been added together like this:

200 + 400 = 600
200 + 0 = 200

and so we have ended up at point (600,200). The same rules apply when using the MOVER command. If we entered the commands:

```
CLG
MOVER 150,200
MOVER 100,150
```

then if our maths are correct the cursor would end up at point (250,350).

So you see it is just as easy to use relative coordinates as it is to use absolute ones. The only problem is that you can easily wander off the edge of the screen when using relative coordinates, so you need to keep track of exactly where you are moving the cursor to.

You might like to try substituting the MOVE and DRAW commands in some of the programs in this article with MOVER and DRAWR. If you use the same coordinates, the shapes produced will differ greatly from the originals, but you should be able to adjust the values in order to restore the correct shapes.

In theory any shape can be drawn using either the absolute or the relative coordinate system. There are two points to remember. First of all, negative coordinates are allowed, so the commands:

```
CLG
MOVER 200,200
MOVER -200,-200
```

would move the cursor out to (200,200) and back again to (0,0). In fact negative numbers are almost a necessity when drawing with relative cooordinates.

ZOOM TECHNIQUE IS VERY HANDY

Program IX demonstrates this point by drawing a square using relative coordinates.

The second point to bear in mind is that it is possible to move the cursor off the screen by using cooordinates greater than those normally available.

We have so far only used co-ordinates within the range (639,399), but it is quite legal, and sometimes necessary, to specify numbers greater than these.

Enter the commands:

```
CLG
DRAW 1000,1000
```

and the computer will quite happily draw a line to an imaginary point (1000,1000). Of course the line will disappear off the screen. To prove that the cursor has in fact gone to the specified point enter another command:

```
DRAW 500,0
```

and you will see that the line reappears as though it had in fact come from point (1000,1000).

Program X demonstrates how off-screen coordinates may be used. The program begins by drawing a triangle small enough to fit on the screen. Now if you press the + key the triangle is erased and redrawn slightly larger. Keep on pressing the + key and eventually the triangle will be drawn outside the limits of the screen perimeter. It is in fact being drawn using off screen coordinates.

Use the – key to reduce the triangle down to its original size. This zoom technique is often used to vary the size of images on the screen in order to show more detail in complex drawings.

Anyway, that's enough for this month. Try plotting some shapes of your own with DRAW, MOVE, DRAWR and MOVER. Next month we'll move onto plotting individual points.

```
10 REM PROGRAM IX
20 MODE 1
30 MOVE 300,300
40 DRAWR 200,0
50 DRAWR 0,-200
60 DRAWR -200,0
70 DRAWR 0,200
```

Program IX

```
10 REM PROGRAM X
20 MODE 1
30 colour=1
40 x=100:y=100
50 a=400:b=300
60 c=400:d=100
70 GOSUB 110
80 IF NOT INKEY(28) THEN GOSUB 170
90 IF NOT INKEY(25) THEN GOSUB 250
100 GOTO 80
110 REM draw triangle
120 MOVE x,y
130 DRAW a,b,colour
140 DRAW c,d
150 DRAW x,y
160 RETURN
170 colour=0
180 GOSUB 110
190 x=x-8:y=y-8
200 a=a+8:b=b+8
210 c=c+8:d=d-8
220 colour=1
230 GOSUB 110
240 RETURN
250 colour=0
260 GOSUB 110
270 x=x+8:y=y+8
280 a=a-8:b=b-8
290 c=c-8:d=d+8
300 colour=1
310 GOSUB 110
320 RETURN
```

Program X

ZOOM TECHNIQUE IS VERY HANDY

NOISES!

Part V of NIGEL PETERS' series on coaxing sounds from the CPC464

WE'VE covered a lot of ground in the first four articles of this series, so this month we'll take a little time to reconsider what we've learnt.

The SOUND command is the one that tells the Amstrad to make a noise. Not only that, it controls the type of noise made, how long it lasts and how loud it is.

At its simplest, the structure of the SOUND command is:

```
SOUND channel,pitch,duration,volume.
```

You can hear what it sounds like by keying in:

```
SOUND 1,200,100,7
```

By comparing it to the basic structure, you'll see that the channel parameter is 1. The Amstrad has three sound channels – channel A, channel B, and channel C. Each can make a noise independently of the other, so you can have three different notes playing at the same time.

In the above example the channel parameter was 1, so channel A made the sound. If it had been 2, channel B would have produced the noise, while 4 would select channel C.

There is a lot more to the channel parameter (which we'll come to next month) but for the present we'll just stick to the values 1, 2 and 4 selecting channels A, B and C respectively.

The next parameter we come to controls the pitch, deciding how high or low the note can be. It can take whole number values from 0 to 4095.

The larger the value of the pitch parameter the lower the note produced. Similarly, the smaller its value, the higher the note produced.

When you use a SOUND command you have to give it a channel and a pitch parameter. You can't leave them out as in some of the other parameters we'll meet.

The duration parameter decides how long the note is going to last. It can take integer values ranging ▶

between 32767 and −32768.

The positive values decide how long the note is going to last, measured in hundredths of a second. Hence a duration parameter of 100 will last for one second while 50 will last for half a second and 1000 for 10 seconds.

Negative values of the duration parameter have a different effect. They still determine how long the note is going to last, but in a more roundabout method.

When the number is negative it tells the Amstrad that a volume envelope is to be used to alter that note and that that volume envelope is to be repeated. If the duration parameter is −3 the volume envelope is repeated three times. If it's −20 it's repeated 20 times.

When a negative duration parameter is used the length of the note produced by the SOUND command depends on the number of times the envelope is repeated. Each volume envelope lasts for a fixed time, decided in its definition.

So if a particular volume envelope lasts for two seconds and the duration parameter is −3 then the whole note will last six seconds. The repeated volume envelope determines the length of the whole note.

We'll have more to say about volume envelopes later.

The volume parameter speaks for itself. Normally it can range from 0 (silence) to 7 (loudest) but if a volume envelope is used the range becomes 0 to 15.

There's no difference in the maximum loudness. Without an envelope 7 is just as loud as 15 with an envelope. It's just that with a volume envelope you can have finer control over the loudness.

You don't have to specify either the volume or the duration parameter. If you leave them out the Amstrad assumes that you mean a note of volume 4 with a duration of 20. These are the default values of the parameters.

You should now understand why:

```
SOUND 4,200
```

is the same as

```
SOUND 4,200,20,4
```

As you can see, our basic SOUND command isn't too hard to grasp. In fact you could produce some nice noises using only the four parameters we've covered so far. However if you really want to take full advantage of the CPC464's abilities you have to know a little about the volume and tone envelopes.

Now our SOUND command takes the structure:

```
SOUND channel,pitch,duration,volume,
volume envelope,tone envelope
```

As you can see, we've added two more parameters. The volume envelope parameter can take values between 1 and 15. These numbers refer to previously defined volume envelopes.

A parameter of 2 will bring volume envelope number 2 into play while a parameter of 5 invokes the volume envelope labelled number 5. These volume envelopes determine how the loudness of the note produced by a SOUND command varies as the note plays.

There is a volume envelope 0. This is the default envelope and plays for two seconds at the volume specified in the SOUND command.

Having seen how volume envelopes are called by a SOUND command, let's take a brief look at how they're defined. This is done using the ENV command which, at its simplest, takes the form:

```
ENV N,P,Q,R
```

N is just the number labelling the volume envelope, and takes values from 1 to 15. The volume envelope works by changing the loudness of a note in a fixed number of steps, each step lasting for a short period of time.

The P parameter, which ranges from 0 to 127, determines the number of steps there will be in the envelope. The Q parameter, which can take values between −128 and 127, picks the volume change for each of these steps. R decides how long each step will last, measured in hundredths of a second.

So defining a volume envelope with:

```
ENV 12,5,2,100
```

creates envelope 12, which consists of five steps, each lasting one second with the volume increasing by 2 for each step.

You should be able to hear its effect on:

```
SOUND 4,200,500,3,12
```

Table I shows the parameter ranges for the ENV command.

You'll probably remember that our previous envelope definition only referred to one section of a volume envelope. In fact you can define up to five sections in a volume envelope and the corresponding formula is:

```
ENV N,P1,Q1,R1,P2,Q2,R2,
    P3,Q3,R3,P4,Q4,R4,
    P5,Q5,R5
```

The pitch or tone envelope is very similar to the volume envelope except that it affects the highness or lowness of a note, not its loudness. The basic structure of its definition is:

```
ENT S,T,V,W
```

As you might guess, S just labels the envelope. It can take values between 1 and 15. The default envelope has S as 0 and leaves the pitch unchanged.

The T parameter decides on the number of steps in the pitch envelope. Its value can range from 0 to 239.

The change in pitch that occurs at each of these steps is given by V. This ranges from −128 to 127. Notice that a negative V raises the pitch at each step while a positive value lowers it.

The W parameter gives the length of each step. Measured, as usual, in hundredths of a second, it ranges from 0 to 255.

So:

```
ENT 13,10,20,100
```

Parameter	Number N	Number of steps in section P	Volume change per step Q	Time length of each step R
Range	1 to 15	0 to 127	−128 to 127	0 to 255

Table I: Parameter ranges for ENV command

defines pitch envelope number 13. This has 10 steps, each lasting for one second. At each step 20 is added to the pitch parameter of the SOUND command. This results in a note descending in pitch.

Listen to its effect on:

```
SOUND 2,500,1000,7,0,13
```

Like the volume envelope before it, the pitch envelope can have up to five sections, each using the above parameters. This more comprehensive definition is:

```
ENT S,T1,V1,W1,T2,V2,W2,
    T3,V3,W3,T4,V4,W4,
    T5,V5,W5
```

Table II shows the parameter ranges for the ENT command.

The pitch envelope can be made to repeat simply by making the label negative. Listen to the effect that:

```
ENT -5,5,-20,100
```

has on

```
SOUND 1,600,2500,7,0,5
```

The pitch envelope lasts for five seconds, then repeats itself as the noise keeps on going. Notice that while the label number in the definition is negative, the SOUND command still calls it with a positive number.

Also notice that, unlike the volume envelope, the repeated pitch envelope has no effect on the duration of the noise.

When the duration in the SOUND command ends the note stops, even halfway through a repetition. Listen to what happens when:

```
ENT -14,3,100,50
```

has its wicked way on:

```
SOUND 2,1000,400,7,0,14
```

Before we take our leave of the envelopes notice that by themselves ENV and ENT are mute. They don't make a noise, they just affect the noises made by SOUND commands.

Parameter	Number S	Number of steps in section T	Pitch change per step V	Time length of each step W
Range	1 to 15 (-ve for repeat)	0 to 239	−128 to 127	0 to 255

Table II: Parameter ranges for ENT command

As you can see from the above, we've come a long way in four articles. Our simple SOUND command has grown from four fairly obvious parameters to six, with ENT and ENV thrown in for good measure!

However I think you'll agree that if you take them step by step they're a lot simpler than they look at first glance.

And now I'll add one last parameter to the SOUND statement. It's the parameter that tags onto the end and allows us to make noise. Yes, I know that's all we've been doing for the past four months, but this is a rather different noise. Try:

```
SOUND 2,300,1000,7,0,0,13
```

and hear the effect. Not what you might have expected.

The explanation lies in the last parameter tagged on the end of the SOUND command. This is the aptly named noise parameter. It can take values from 1 to 15 (0 is the default and does nothing) adding a variety of semi-random noises to the notes you might expect.

Now our SOUND command looks like:

```
SOUND channel,pitch,duration,volume,
volume envelope,tone envelope,noise
```

Try experimenting with the noise parameter to see what it can do. Use a note like:

```
SOUND 1,200,200,7
```

and then add noise to it with lines such as:

```
SOUND 1,200,200,7,0,0,1
```

and

```
SOUND 1,200,200,7,0,0,15
```

Use all the noise values between 1 and 15 and then try it out with pitch and volume envelopes and different pitch and duration parameters. You'll come across all sorts of marvellous sound effects. *(Send them in to Postbag if you've got any you want to share.)*

Program I is my version of a steam engine.

And that's it for this month. I've summed up all our SOUND parameters in Table III and leave you to experiment with noise and noises.

One last point. Remember from the first article I told you to get rid of any odd sounds by entering:

```
SOUND 129,100,0,0
```

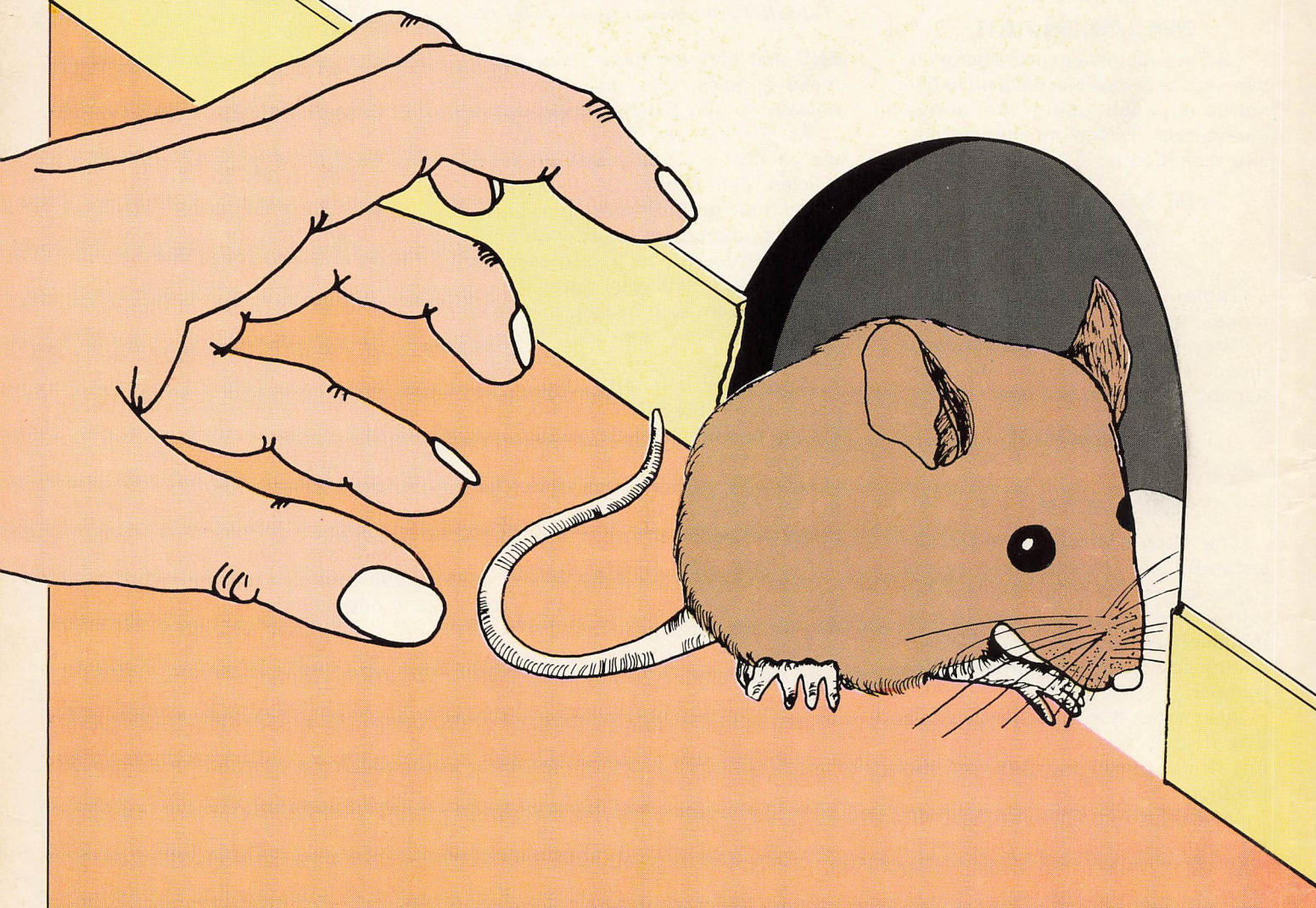
into your Amstrad? What does a channel parameter of 129 mean? We'll be coming to that next month.

```
10 REM Program I
20 REM Steam Engine
30 pitch=1
40 FOR duration=100 TO 20 STEP -10
50 SOUND 1,pitch,duration,7,0,0,1
60 SOUND 1,pitch,duration,7,0,0,15
70 NEXT duration
80 WHILE NOT true
90 SOUND 1,pitch,duration,7,0,0,1
100 SOUND 1,pitch,duration,7,0,0,15
110 WEND
```

	Channel	Pitch	Duration	Volume		Volume envelope	Pitch envelope	Noise
				without envelope	with envelope			
Range	1=A 2=B 4=C	0 to 4095	32767 to −32768	0 to 7	0 to 15	1 to 15	1 to 15	1 to 15
Default	none	none	20	4	12	0	0	0

Table III: Parameter ranges for SOUND command

Pounce on a



VARIABLES

ad2,3	Machine code addresses for subroutines.
s%	Score.
t%	Best score.
x% y%	Coordinates of man.
g% h%	Coordinates of mouse.
mou	Ascii values of mouse character.
man	Ascii values of man characters.
dir	Mouse direction indicator.
i% j%	Coordinate changes for screen locations to be tested.
k% m%	Mouse direction change indicators.
x$	Inkey variables for responses.
n	Loop counter.
tit	Ascii data read for title.
in	Keyboard or joystick inputs.
b,c,d	Machine code routine variables.
ch	Ascii value of screen location.

WINDOWS

# 0	Whole screen.
# 1	Mouse title.
# 2	Score and best score.
# 3	"GOT HIM" window, when mouse is captured.
# 4	Main graphics area.

MOUSE !

Get to grips with GRAHAM REDMAN's challenging chase game

MOUSE is a very simple graphics game written mainly in Basic in which you chase a mouse, gradually cornering him in order to pounce.

The mouse moves continuously in a diagonal direction over the whole screen, bouncing off the walls.

Using the joystick or cursor keys you can move yourself around the screen, but as you do you leave a trail of "blocks" behind which act like a wall and will confine the mouse so you can catch it.

To do this all you have to do is place yourself over the mouse. Of course the lowest score is the best in this game, so the secret is to act quickly and pounce. My record so far is 15.

Although this is a Basic program there are two very simple machine code routines. One returns the Ascii value of an input through the keyboard or joystick and the other returns the Ascii value of a character at a particular screen location.

The program is written entirely in character (low resolution) graphics in Mode 1 and uses LOCATE and PRINT for movement.

The two mouse characters and the wall "block" are defined characters (from Ascii 201), while the man characters are from the Amstrad Ascii set. Each time the mouse moves it tests the Ascii value of two squares ahead and to one side of it (see Figure 1). It has to do this in order to know which way to turn when it detects a wall. If, however, it comes to a character diagonally in front it will wipe it out and go through the gap.

Having an unpredictable mouse adds a new dimension to the game, hence the warning in the introduction – "BEWARE: THIS MOUSE CAN GNAW".

Mouse is well documented with REM statements and should not be too difficult to follow.

In order to get quick keyboard response minimum speed key values are used on line 50. However this can cause problems during programming and testing if a break or error occurs due to de-bounce.

To overcome this a break is trapped in order to restore the normal speed key values at line 1350.

Errors are not trapped, so their source can be examined, but the small Enter key has been redefined (line 40 in the program) so that if it is pressed once it will restore normal working of the keyboard.

In the case of a syntax error the large Enter key must be pressed first in order to come out of the edit. Then the small Enter key can be pressed and the edit can then be done.

If all this seems complicated then the best idea is to miss out the SPEED KEY 1,1 on line 50, enter the whole program, debug it, then complete line 50.

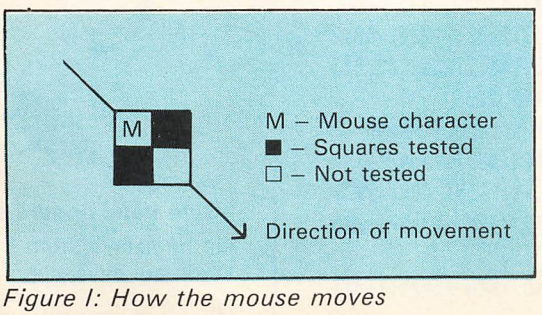
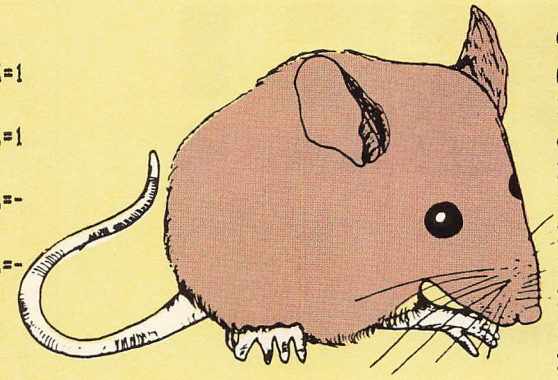


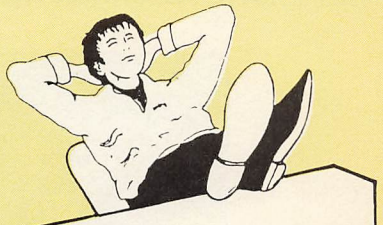
Figure 1: How the mouse moves

```
10 REM*****MOUSE BY G.REDMAN*****
20 REM(c)Computing with the Amstrad
30 REM*****INITIAL SETTING UP*****
40 ON BREAK GOSUB 1350:KEY 139,"SPEED
KEY 10,5:INK 1,25:PEN 1"+CHR$(13)
50 RANDOMIZE TIME:SPEED KEY 1,1
60 ad2=34999:GOSUB 1080:ad3=35020:GOSU
B 1240
70 REM**MOUSE & MARKER CHARACTERS**
80 SYMBOL AFTER 200:SYMBOL 201,0,8,60,
126,34,12,0,0
90 SYMBOL 202,0,16,60,126,68,48,0,0
100 SYMBOL 203,0,0,60,60,60,60,0,0
110 REM*COLOUR AND WINDOW SETTINGS*
120 BORDER 4:INK 0,0:INK 1,0:INK 2,0:I
NK 3,0:PAPER 0:CLS
130 WINDOW#2,2,39,25,25:WINDOW#4,2,39,
2,23:PAPER#4,0:PAPER #2,2:PEN#2,0
140 t%=10000
150 GOSUB 750:REM**TITLE & INSTR***
160 REM***DRAW BORDER***
170 CLS:INK 3,0:PEN 3:PRINT STRING$(40
,143)
180 FOR n=2  TO 25:LOCATE 40,n:PRINT C
HR$(143);:NEXT
190 LOCATE 1,24:PRINT STRING$(40,143)
200 LOCATE 1,1:FOR n=1 TO 25:LOCATE 1,
n:PRINT CHR$(143);:NEXT:INK 3,6
210 REM***SCORE WINDOW***
220 CLS#2:LOCATE#2,3,1:PRINT#2,"Your S
core":s%=0:LOCATE#2,20,1:PRINT#2,"Best
 Score":LOCATE#2,30,1:IF t%<>10000 THE
N PRINT#2,t%
230 REM*SOUND VARIABLES & START**
240 ENT 1,1,12,6,1,-12,6:ENV 1,6,-1,1
250 x%=INT(RND(1)*30)+4:y%=INT(RND(1)*
20)+3:g%=INT(RND(1)*30)+4:h%=INT(RND(1
)*20)+3:dir=INT(RND(1)*4)+1:PEN 2
260 REM***MAIN PROGRAM LOOP***
270 IF dir=1 OR dir=2 THEN mou=202 ELS
E mou=201
280 LOCATE g%,h%:PRINT CHR$(mou)
290 PEN 0:LOCATE#2,13,1:PRINT#2,s%:s%=
s%+1
300 GOSUB 950:REM**MOVE MAN**
310 LOCATE g%,h%:PRINT CHR$(143):PEN 2
320 GOSUB 400:REM**MOVE MOUSE**
330 LOCATE (g%+i%),h%:GOSUB 1190:REM**
LOOK FOR WALL**
340 IF k%=1 THEN m%=1:GOSUB 460:REM**C
HANGE MOUSE DIRECTION**
350 m%=0:k%=0:LOCATE g%,(h%+j%):GOSUB
1190
360 IF k%=1 THEN m%=2:GOSUB 460
370 k%=0:m%=0:GOTO 270
380 REM***END OF MAIN LOOP***
```

```
390 REM***MOVEMENT ROUTINE***
400 IF dir=1 THEN g%=g%+1:h%=h%+1:i%=1
:j%=1
410 IF dir=2 THEN g%=g%+1:h%=h%-1:i%=1
:j%=-1
420 IF dir=3 THEN g%=g%-1:h%=h%-1:i%=-
1:j%=-1
430 IF dir=4 THEN g%=g%-1:h%=h%+1:i%=-
1:j%=1
440 RETURN
450 REM**DIRECTION CHANGE ROUTINE**
460 IF x%=g% AND y%=h% THEN GOTO 570
470 SOUND 1,160,6,15,1,1,0
480 IF dir=1 AND m%=1 THEN dir=4:RETUR
N
490 IF dir=1 AND m%=2 THEN dir=2:RETUR
N
500 IF dir=2 AND m%=1 THEN dir=3:RETUR
N
510 IF dir=2 AND m%=2 THEN dir=1:RETUR
N
520 IF dir=3 AND m%=1 THEN dir=2:RETUR
N
530 IF dir=3 AND m%=2 THEN dir=4:RETUR
N
540 IF dir=4 AND m%=1 THEN dir=1:RETUR
N
550 IF dir=4 AND m%=2 THEN dir=3:RETUR
N
560 REM***MOUSE CAPTURED ROUTINE***
570 IF x%>36 THEN x%=36
580 IF y%>20 THEN y%=20
590 IF x%<3 THEN x%=3
600 IF y%<3 THEN y%=3
610 WINDOW#3,(x%-2),(x%+2),(y%-2),(y%+
2):PAPER#3,3:CLS#3:PEN#3,2:LOCATE#3,2,
2:PRINT#3,"GOT":LOCATE#3,2,4:PRINT#3,"
HIM"
620 ENT 1,15,7,2,15,-7,2
630 FOR n=100 TO 250 STEP 2:SOUND 1,n,
3,6,0,1,0:NEXT
640 SOUND 1,210,50,6,0,0,9
650 IF s%<t% THEN t%=s%-1
660 LOCATE #2,30,1:PRINT #2,t%
670 LOCATE 5,24:PEN 1:PRINT"  PRESS SP
ACE BAR WHEN READY   "
680 WHILE INKEY$<>" ":WEND
690 REM****REPEAT GAME****
700 CLS#4:LOCATE 5,24:PEN 3:PRINT STRI
NG$(31,143):PEN 2:LOCATE 12,13:PRINT "
Another game ? (Y/N)"
710 x$=UPPER$(INKEY$):IF x$="Y"THEN CL
S #4:GOTO 220
720 IF x$="N" THEN GOTO 1350
730 GOTO 710
740 REM****TITLE & INSTRUCTIONS****
750 WINDOW#1,8,33,2,9:PEN#1,1
760 RESTORE 900
```



```
770 FOR n=1 TO 104
780 READ tit:PRINT#1,CHR$(tit);
790 NEXT
800 PEN#1,2:PRINT#1:FOR n=1 TO 13:PRIN
T#1,CHR$(202);CHR$(201);:NEXT n
810 PEN#4,3:LOCATE#4,1,8:PRINT#4,"  U
sing the joystick or the cursor  keys
 you  must  chase  the  mouse ";:PEN#4
,2:PRINT#4,CHR$(202);;:PEN#4,3:PRINT#4
,"and capture him."
820 PRINT #4:PRINT#4,"  To  do  this
 you  must  move  yourman  to  the sa
me position  as  the  mouse. The  trai
l  you  leave  behindwill act like a
wall and  confine  themouse so you can
 capture him."
830 PRINT#4:PRINT#4,"  Do this as  fa
st  as  you can. The LOWEST score is t
he best."
840 PEN 2:LOCATE 5,23:PRINT "BEWARE, T
HIS MOUSE CAN GNAW !!!!"
850 LOCATE 10,25:PRINT "PRESS THE SPAC
E BAR.";
860 INK 1,14:INK 2,24:INK 3,15
870 WHILE INKEY$<>" ":WEND
880 CLS#4:RETURN
890 REM**DATA FOR TITLE******
900 DATA 214,32,32,32,32,215,32,32,214
,143,215,32,32,214,32,215,32,32,214,14
3,215,32,32,214,143,215
910 DATA 143,215,32,32,214,143,32,32,1
43,32,143,32,32,143,32,143,32,32,213,2
10,32,32,32,143,210,32
920 DATA 143,213,215,214,212,143,32,32
,143,32,143,32,32,143,32,143,32,32,32,
208,215,32,32,143,208,32
930 DATA 213,32,213,212,32,212,32,32,2
13,143,212,32,32,213,143,212,32,32,213
,143,212,32,32,213,143,212
940 REM***INKEYS/JOY m/c ROUTINE***
950 CALL ad2+1:in=PEEK(ad2):man=248
960 PEN 1:LOCATE x%,y%:PRINT CHR$(203)
970 IF in=8 OR in=242 THEN x%=x%-1:man
=250
980 IF in=9 OR in=243 THEN x%=x%+1:man
=251
990 IF in=11 OR in=240 THEN y%=y%-1:ma
n=249
1000 IF in=10 OR in=241 THEN y%=y%+1:m
an=249
1010 IF x%>39 THEN x%=39
1020 IF x%<2  THEN x%=2
1030 IF y%>23 THEN y%=23
1040 IF y%<2 THEN y%=2
1050 PEN 2:LOCATE x%,y%:PRINT CHR$(man
):PEN 0:SOUND 1,200,1,2,0,0,0
1060 RETURN
1070 REM***POKE INKEYS m/c***
1080 c=INT(ad2/256):b=ad2-256*c
1090 RESTORE 1160
1100 FOR n=ad2 TO ad2+13
1110 READ d:IF d=999 THEN d=b
1120 IF d=998 THEN d=c
1130 POKE n,d
1140 NEXT n
1150 RETURN
1160 DATA 0,62,0,50,999,998,205,27
1170 DATA 187,208,50,999,998,201
1180 REM**CHECK CHAR POSITION**
1190 CALL ad3+1
1200 ch=PEEK(ad3)
1210 IF ch=143 OR ch=203 THEN k%=1
1220 RETURN
1230 REM**POKE CHAR CHECK m/c***
1240 c=INT(ad3/256):b=ad3-256*c
1250 RESTORE 1320
1260 FOR n=ad3 TO ad3+13
1270 READ d:IF d=999 THEN d=b
1280 IF d=998 THEN d=c
1290 POKE n,d
1300 NEXT n
1310 RETURN
1320 DATA 0,62,244,50,999,998,205,96
1330 DATA 187,208,50,999,998,201
1340 REM*****END ROUTINE*******
1350 CLS:INK 1,25:PEN 1:LOCATE 1,13:PR
INT "Type R to RUN again or E to END p
rogram.":SPEED KEY 10,5
1360 x$=UPPER$(INKEY$):IF x$="E" THEN
END
1370 IF x$="R" THEN RUN
1380 GOTO 1360
```



Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

ALAN McLACHLAN investigates ways of getting sick programs back on their feet

FIRST of all a big thank you to everyone who has written in during the last few weeks in connection with my faster circle request.

You may recall that in March's article I dabbled at a spot of programming and produced what I considered a rather nifty little program. It involved three rotating elipses and I asked if anyone could make the program draw them any faster.

I had a variety of first class replies and we included the first one received in last month's Postbag. Many others were interesting, touching on ideas that Kevin Edwards has used in this issue's animation article.

Last month I introduced you, tongue in cheek, to the host of people who write to us with programs that they have typed in and cannot get to run. (By the way I forgot to mention the hairdresser . . . *"I've been through it with a fine toothcomb".)* Inevitably these people have made typing errors and some actually admit it (see Mr Bellis's letter on Page 73 of last month's issue.)

In the same article I promised you some useful hints to help get your sick programs back on their feet again and I'm going to make a start now.

I make no apologies for re-emphasising that the main reason for the majority of your typed in programs not running is that somewhere you have made a mistake. No, not in buying the your micro in the first place, although you may feel like chucking it through the window sometimes!

You have either mistyped or omitted some item, or have inserted something in the program that is not only not required, but whose presence is fouling up the program.

These typing errors are certainly responsible for the variety of error messages that will greet you, ranging from a simple "Syntax error" to the heart-stopping "Subscript out of range". No, I don't relish this second one either.

Preventing the typing errors rather than curing the results would seem to be the ideal solution. It should be easy enough. Type in the listings slowly, carefully, checking each line once it has been entered. You can even move the cursor across the line to highlight each letter or to count each data entry as each line is finished.

Tedious? Dead right, but I did it on many occasions in my early days in computing, and although I was fed up to the back teeth, my programs worked. Believe me, that makes up for all the tedium.

But, quite naturally, people get impatient and prefer to crash on regardless in a rush to get the tedious part finished. Then they start looking for errors when RUN doesn't work. If this is the way you prefer to do it, we must look at some ways to make those errors easier to find.

A good idea is to make use of one of the features of your Amstrad. Why not type the listing in using lower case letters (except of course where upper case ones are necessary in PRINT or DATA statements)? If you haven't already realised, your micro translates all commands entered in lower case into upper case when the program is RUN or LISTed.

This has a bonus in that any incorrectly entered command will remain in lower case – standing out like a sore thumb when you list the line. Try it – type in Listing I exactly as it is. Don't correct any errors.

Run it and see for yourself. The program is intended to clear the screen in Mode 1, print the text at location 2,2 count to 1000, then wipe out the text with spaces. Line 80 is a dummy to prevent the "Ready" prompt appearing.

The first error message you'll get is "Syntax error at line 10" and with it your Amstrad lists the line and automatically puts you into Edit mode. This error is probably easy to spot – the command *mode* is still in lower case because I forgot the space between it and the number 1. Correct the error and run the program again.

If you carry on correcting each error as it appears you should realise how useful this idea can be. By the way, the only mistake which is not a "Syntax error" puts a 0 on the screen at location 2,2. Have you any idea why? Think about it and I'll tell you the answer at the end of the article.

It is important if you are going to enjoy your hobby completely that very early on you become reasonably proficient at picking your way through a listing by just reading it. In fact, it is imperative if you want to be able to debug your own programs. It's as important as a musician being able to read music without actually playing it.

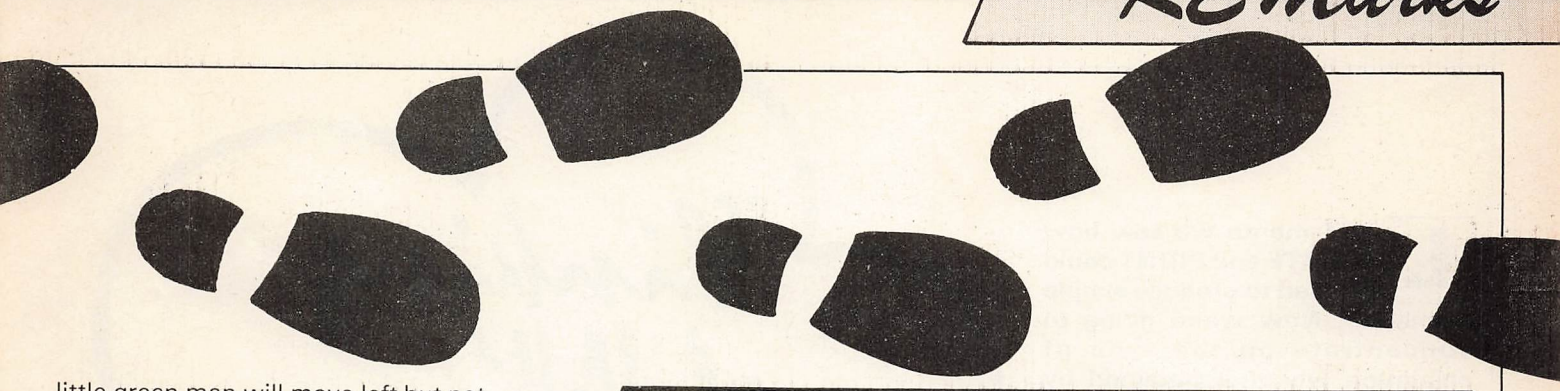
This is why we attach so much importance to REM statements in our magazine listings. They're there for your benefit as well as ours.

We insist that our games writers use lots of them to show the program's structure. Also we ask them to give us lists of subroutines and variables to assist you to find them in the program.

Finding out why your instructions are not working correctly, or why your

```
10 mode1:cls
20 pen 2
30 loacate 2,2
40 print "Al is the greatest"
50 for x = 1 to 1000:next x
60 locate 2,2
70 pint spc18
80 goto80
```

Listing I



little green man will move left but not right, is easier if you can identify the relevant subroutine. Read through the listing and try and make some sense of what is going on.

Just in case you missed Al's Beat last month (there must be at least one of you) here's a repeat of a simple debugging hint that should make your errors easier to identify. Let's assume you typed in line 30 of Reaction Timer (Page 62, February issue) as:

```
30 MODE 1:INK 0,1:INK 1,24:INK 2,20:
INK 3,6:PEN 1:DIM m$(8):GOSUB100
```

Your micro will respond "Syntax error in 30" and the mistake should be fairly obvious – GOSUB 100 needs a space.

But what if there were more statements in the line and it were not so obvious? A simple hint is to split the line in two as follows:

```
30 MODE 1:INK 0,1:INK 1,24:INK 2,20
31 INK 3,6:PEN 1:DIM m$(8):GOSUB100
```

Now you will get "Syntax error in 31". This can now be split:

```
32 INK 3,6:PEN 1
33 DIM m$(8):GOSUB100
```

and the result will be "Syntax error in 33".

In fact, apart from in just one set of circumstances, you can split any line you want at any colon, and make as many extra lines as the program will allow you.

Gradual elimination will narrow down the alternatives until you are left with the offending statement on its own. Your micro can't tell you what's wrong with it, but with careful checking and perhaps experiment you should come up with the solution.

The circumstances under which you have to be extremely careful are when the contents of the line are all governed by an IF statement. To be on the safe side you must leave this line complete.

However in desperation you can split it. But be warned. Your program almost certainly will not run correctly,

'It is important to become reasonably proficient at picking your way through a listing by just reading it'

although any syntax or similar error should be spotted.

There are a couple of other useful techniques that will help you keep track of where your program is going or where it has got to. The first is the use of the command STOP.

By inserting this at a strategic point in your program you can check to see if the program is working correctly. When it encounters STOP your program will halt with a message "Break at line XXX" when it reaches XXX, the line you've chosen to insert the command.

If all is well, you know the error is beyond this point in the program. If, however, all is not well – perhaps it hasn't even stopped – you'll have to backtrack from this line to see what's amiss. This is extremely helpful in identifying a line which may be putting something on the screen incorrectly.

This technique is ideal for sorting out any graphics errors that may have crept into your listing. You can spot these most of the time by the way the screen looks.

Let's assume you have a character displayed that should be a man and in fact it has the appearance of nothing more than a shapeless splodge.

No one can teach you where to put the STOP command. You'll probably have to establish this by trial and error. If, however, you've followed my advice and worked out which way the program's going from the REM statements and accompanying notes, you'll know roughly where it should go.

In this case once you've established the correct place, the line before the "Break" may contain a statement such as:

```
PRINT CHR$(243)
```

This in turn should point you to the subroutine which defines SYMBOL 243. You have probably entered the data incorrectly in this subroutine.

Another useful tip is the use of the beep produced by:

```
PRINT CHR$(7)
```

This command, correctly inserted, can tell you whether a particular part of the program is being reached or not. It can also indicate how many times you have been through a loop if there is any doubt at all. This is particularly useful if you are completely in the dark as to where your program is going. You can hear the beep and you know which line you put it in.

Armed with these three hints you should be able to make some inroads into that program that just refuses to budge. You may not know *what* your error is but at least you'll know *where*.

I think that's enough to be going on with, but before I sign off, a word to anyone who may be slightly confused having tackled Michael Noel's last graphics article – any more slips like this and it will be his last!

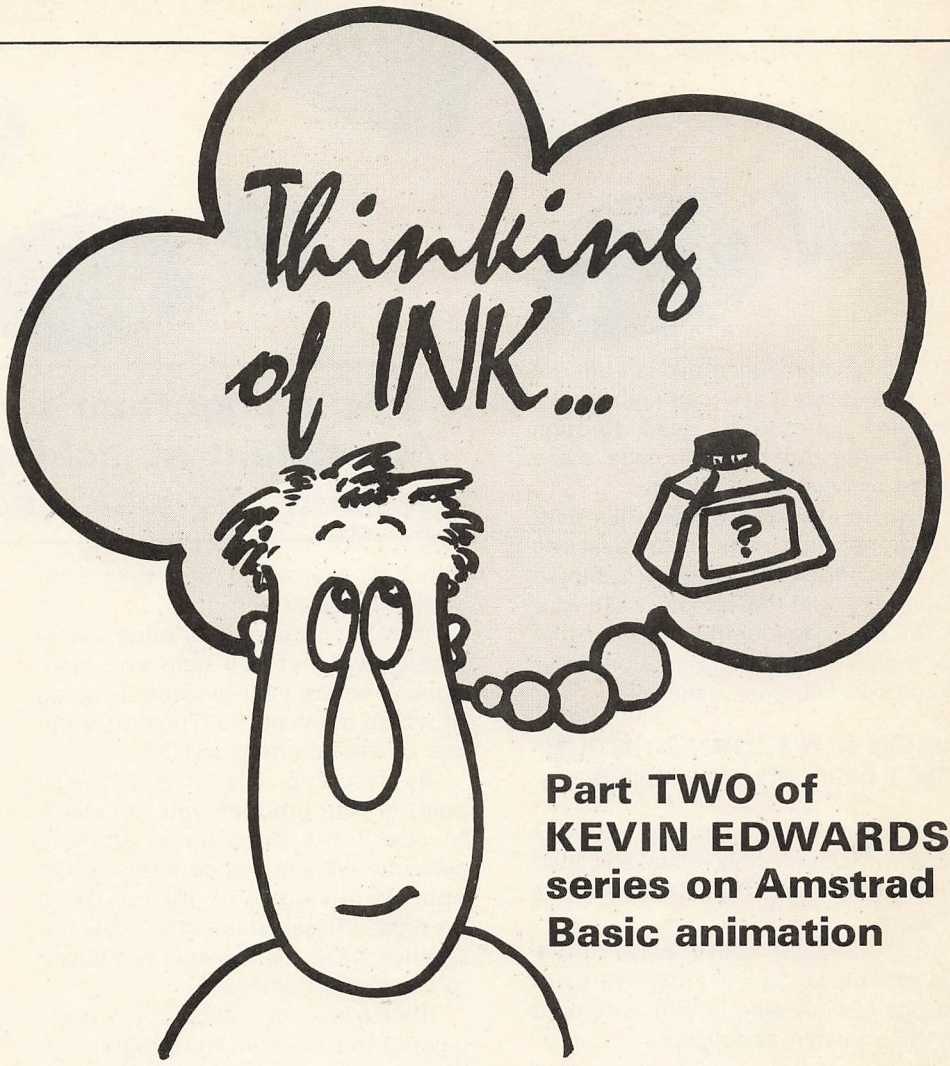
His explanation of the coordinate system for establishing text windows appeared to contradict itself at one point. Suffice it to say that his second statement that "column" comes before "row" as in CR (Carriage Return) was correct.

Oh! nearly forgot. The answer to the Listing I problem is that "spc18" in line 70 should have brackets around the number like this:

```
70 print spc(18)
```

However the micro accepts the wrong entry as a variable to which nothing has been assigned. It therefore prints a 0 at location 2,2 on the screen. If you spotted that, well done!

See you next month.



Part TWO of KEVIN EDWARDS' series on Amstrad Basic animation

LAST month we saw how LOCATE and PRINT could be used to produce simple animation. Now we're going to concentrate on the type of animation which is obtained by clever use of the INK command.

As you already know, the CPC464 has three different graphic modes – 0, 1 and 2 – each with its own unique characteristics. For example, Mode 0 can have up to 16 colours on screen at any time, Mode 1 has four colours and Mode 2 has two colours.

These are just selections of the colours available. Each mode can use any combination of the 27 colours shown in Table I.

However despite the fact you can use any of the 27 colours in any mode the maximum number of colours on the screen at any one time always remains the same for each mode. What you're doing is picking 2, 4 or 16 colours from a palette of 27.

Not only can you choose which colours you have on screen, you can also swap any one colour for another. For example, if you have a blue circle on the screen you can change it to red with one simple Basic command:

```
INK X,Y
```

where X is the colour number to be changed (blue in the example) and Y is the new colour (red in the example).

If you still don't understand how INK works see Michael Noels' excellent Graphics article in the February issue of *Computing with the Amstrad.*

The example programs this month all use Mode 0, the 16 colour mode.

Ink Number	Colour		
0	Black	13	White
1	Blue	14	Pastel Blue
2	Bright Blue	15	Orange
3	Red	16	Pink
4	Magenta	17	Pastel Magenta
5	Mauve	18	Bright Green
6	Bright Red	19	Sea Green
7	Purple	20	Bright Cyan
8	Bright Magenta	21	Lime Green
9	Green	22	Pastel Green
10	Cyan	23	Pastel Cyan
11	Sky Blue	24	Bright Yellow
12	Yellow	25	Pastel Yellow
		26	Bright White

Table I: Ink colour numbers

You'll notice that at the end of the first three programs are the commands:

```
MODE 1
CALL &BC02
PEN 1
PAPER 0
END
```

Triggered by the space bar, all they do is restore the colours to their original state at switch on. This is needed because the example programs change the colours to the background colour, blue. And blue writing on a blue background is very hard to read!

Program I remedies the situation. It puts 10 different coloured balls in the middle of the screen. This is done by lines 40 to 90.

However if you stopped the program at line 90 you'd never see them. Each ball is invisible because line 60 changes the colours in pens 1 to 10 to blue (ink 1). The balls are drawn, one after the other, the first ball with pen 1, the second with pen 2 and so on.

However each of these pens has been filled with blue ink and, since blue is the background colour, nothing appears on the screen. The balls blend in, but believe me, they are there.

Lines 100 to 170 are responsible for the animation. They achieve this by revealing a ball, hiding it again and revealing the one next to it.

The variable *dispball* contains the number of the ball being displayed,

```
10 REM PROGRAM I
20 ON ERROR GOTO 180
30 MODE 0
40 LOCATE 1,14
50 FOR x=1 TO 10
60 INK x,1
70 PEN x
80 PRINT CHR$(231);CHR$(32);
90 NEXT
100 dispball=1
110 FOR ball=1 TO 10
120 IF ball=dispball THEN INK ball,4 E
LSE INK ball,1
130 FOR delay=1 TO 10:NEXT delay
140 NEXT ball
150 IF INKEY(47)=0 THEN GOTO 180
160 dispball=dispball+1
170 IF dispball>10 THEN GOTO 100 ELSE
GOTO 110
180 MODE 1
190 CALL &BC02
200 PEN 1
210 PAPER 0
220 END
```

Program I

```
10 REM PROGRAM II
20 ON ERROR GOTO 160
30 DEG
40 MODE 0
50 FOR angle=0 TO 359
60 MOVE 320,200
70 DRAW 200*SIN(angle)+320,175*COS(ang
le)+200,INT(angle/36)+1
80 NEXT
90 dispsegment=1
100 FOR segment=1 TO 10
110 IF segment=dispsegment THEN INK se
gment,4 ELSE INK segment,1
120 NEXT segment
130 IF INKEY(47)=0 THEN GOTO 160
140 dispsegment=dispsegment+1
150 IF dispsegment>10 THEN GOTO 90 ELS
E GOTO 100
160 MODE 1
170 CALL &BC02
180 PEN 1
190 PAPER 0
200 END
```

Program II

```
10 REM PROGRAM III
20 ON ERROR GOTO 180
30 DEG
40 MODE 0
50 FOR fudge=0 TO 9
60 FOR angle=fudge*2 TO 359 STEP 20
70 MOVE 320,200
80 DRAW 200*SIN(angle)+320,175*COS(ang
le)+200,fudge+1
90 NEXT angle
100 NEXT fudge
110 dispstar=1
120 FOR star=1 TO 10
130 IF star=dispstar THEN INK star,4 E
LSE INK star,1
140 NEXT star
150 IF INKEY(47)=0 THEN GOTO 180
160 dispstar=dispstar+1
170 IF dispstar>10 THEN GOTO 110 ELSE
GOTO 120
180 MODE 1
190 CALL &BC02
200 PEN 1
210 PAPER 0
220 END
```

Program III

ranging between 1 and 10. 1 is the leftmost ball and 10 is the rightmost.

The program selects the first ball by initialising variable *dispball* to 1. It is this ball, indicated by the variable *dispball,* which is changed to magenta. This happens because the ink in pen 1 is changed to ink 4, magenta.

When the ball was originally drawn pen 1 was filled with blue ink. Now line 120 has filled it with magenta ink. The result is that we see one magenta ball.

Notice that as the loop formed by lines 100 and 140 cycles, all of the other balls are left in the background colour, blue. The ink is only changed in the first pen, and so only the first ball appears in magenta.

Line 130 provides a short delay and line 150 checks to see if the space bar is pressed. If it is the program ends and restores the colours (lines 180-210). Otherwise line 160 selects the next ball to be displayed by adding 1 to *dispball.* This new ball is to the right of the previous one.

Line 170 checks to see if the previous ball is at the end of the row. If it is, it starts again at the first ball by changing *dispball* to 1 (GOTO 100).

Remember, only one ball is displayed on the screen at one time by the FOR . . . NEXT loop. Line 120 ensures this by making *dispball* magenta and turning all the others blue.

This is a bit off overkill as, if you think about it, only one ball needs switching from magenta back to the background colour. However it does make the programming easy.

The end result of all this is a ball moving across the screen from left to right. Actually you're seeing 10 balls, one after the other, each in a slightly different place. As the balls flash magenta and then re-enter the background colour one after the other, the eye is fooled into thinking that there is only one ball.

It's quite clever when you consider the fact that the animation section is produced without PRINTing to the screen!

To move the ball from right to left, change the following lines:

```
100 dispball=10
160 dispball=dispball-1
170 IF dispball<0 THEN GOTO 100 ELSE
GOTO 110
```

Program II uses INK in a similar way to Program I. This time it changes the colour of segments in a circle. The result is a rotating segment – rather like a windmill with only one sail.

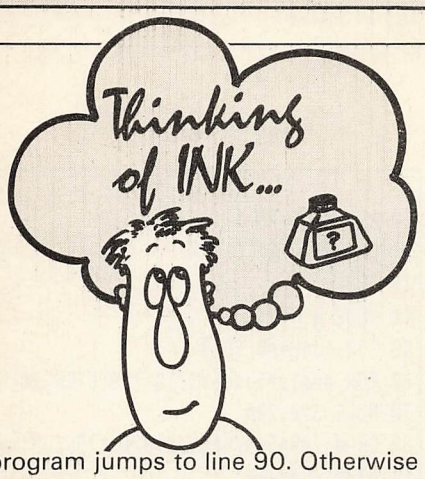
Lines 30-80 draw a solid circle in the middle of the screen. The circle consists of 10 segments, each in a different colour. The segments are numbered 1 to 10 going clockwise in steps of 36 degrees from the top of the circle. The variable *dispsegment* indicates which one of the 10 segments is to be displayed – rather like *dispball* in Program I.

All of the remaining segments are changed to the background colour, blue.

Line 90 initialises *dispsegment* to 1, picking the first segment, and lines 100 to 120 turn all of the segments into the background colour except for the segment indicated by *dispsegment.*

Line 130 tests the space bar. If it is pressed the program will finish.

Line 140 increments the segment being displayed. Line 150 checks to see if all 10 segments have been displayed. If they have, the first segment is selected again and the



program jumps to line 90. Otherwise the jump is to line 100.

As you can see, the end result is very impressive.

Type in and RUN Program III. It's another demonstration of palette switching. By this I mean, changing the colours of the inks in the pens.

The program draws a large star with pen 1. Then it draws another star, this time with pen 2, rotated clockwise by two degrees. This is repeated until 10 different stars have been drawn, each using a different pen.

When this has been completed the stars are displayed, one after the other, by using the same procedure as Programs I and II. The inks filling each pen in turn are toggled between ink 4 and ink 1 – that is, between magenta and the background colour.

I'll let you find out how the program works. It's very similar to the other two so it shouldn't be too difficult. The more advanced among you can have a go at rotating the star anti-clockwise.

Program IV is a quicker solution to the problem posed in Al's Beat in the March issue of *Computing with the Amstrad.*

SINe and COSine functions take a long time to calculate their results. The same calculations using these long winded functions were repeated in the original program. That's why it was so slow. Line 41 solves this and is responsible for most of the increased speed.

And that's it for this month. To test your understanding, why not try changing Programs I to III? Must it always be a blue background? Must the balls always be magenta? And why not have two balls of different colours going in different directions?

There's lots to do. Bye for now.

```
10 MODE 0
12 CALL &BC02
15 DEG
20 ORIGIN 320,200
30 MOVE 0,100
40 FOR i%=0 TO 360
41 s=SIN(i%):c=COS(i%)
50 PLOT s*150,c*150,i% MOD 15 +1:PLOT
s*75,c*150:PLOT s*150,c*75
80 NEXT
100 FOR i%=0 TO 15
110 INK i%,0
115 NEXT
120 WHILE 1=1
130 FOR i%=1 TO 15
140 INK i%,26
150 FOR x=1 TO 100:NEXT
160 INK i%,0
170 NEXT
180 WEND
```

Program IV

Get up to some tricks with your micro ...

... as Aleatoire explains the answer to last month's card coding problem

LAST month I left you pondering the four card coding trick. Here's how it works:

A single card is chosen from five random cards. You push the remaining four cards under a door and a friend in that room names the chosen card – to the consternation of all!

Like many conjuring tricks you have to do most of the work to make it look as though your friend is psychic. In this case you obviously have to tell him which of the unknown 48 cards was chosen.

You do this by the order in which you stack the four cards – remember there are 24 different ways – plus whether they are passed face up or face down. That's the principle.

The program listed on the next page was written to do this trick. Note that the cards are input by just two characters indicating value and suit. Thus the Ace of Clubs becomes AC, the King of Spades becomes KS, and so on.

First let's do a test. Assume the cards you have are Ace of Clubs, 2 of Clubs, 3 of Clubs and 4 of Clubs. Type in the program and enter the first column in Table I using upper case letters.

If this works then you've almost certainly entered the program correctly. Now compare the result with Table II and you might begin to see how it works.

To do the actual trick it's a lot easier if you have an inconspicuous copy of Table II handy and to lay the five cards out in Bridge ascending order thus Clubs > Diamonds > Hearts > Spades. The trick depends on this order.

When a card is chosen calculate its position in the remaining 48 cards by subtracting the number of cards on its left in the layout from the chosen card's normal position in the full 52 card deck. This is obtained using the equation:

position=face value+(suit × 13)

where suit is 0, 1, 2 or 3 for C, D, H and S respectively.

Enter:	*Prints:*	
AC2C3C4C	5 of Clubs	1
AC2C4C3C	6 of Clubs	2
AC3C2C4C	7 of Clubs	3
4C3C2CAC	2 of Hearts	24
AC2C3C4C*	3 of Hearts	25
4C3C2CAC*	K of Spades	48

Table I

Look this number up in Table II and then enter the values (face and suit) of the remaining four remaining cards (from left to right they are 1, 2, 3 and 4) in the given order. If the number is greater than 24 add a trailing *.

Here is an example. You have sorted five cards thus:

1	2	3	*	4
3C	JD	7H	8H	KS

and the 8 of Hearts has been chosen, which is the 8+26–3=31st card in the remaining deck.

The code for 31 is [2 1 3] plus * so enter the cards like this:

JD3C7HKS*

Note that the King of Spades (whenever chosen) is easy to code. Just reverse the order of the remaining cards and add the *. If, in the above example, the King of Spades were the chosen card, you would enter:

8H7HJD3C*

		*			
1	123	25	13	312	37
2	124	26	14	314	38
3	132	27	15	321	39
4	134	28	16	324	40
5	142	29	17	341	41
6	143	30	18	342	42
7	213	31	19	412	43
8	214	32	20	413	44
9	231	33	21	421	45
10	234	34	22	423	46
11	241	35	23	431	47
12	243	36	24	432	48

Table II

With practice you won't need to lay the cards out so obviously or look up the code. Another improvement – which I haven't done – would be to draw a picture of the selected card on the screen. If anyone can do this send it to Paul Daniels, the television conjurer – he likes computers and maybe he'll buy it.

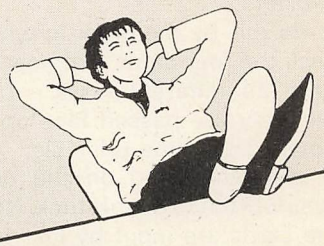
The trick is even less obvious to an observer when it uses a trailing space rather than * to show that the card was >24.

Here's another card trick. Ask someone to lay out 52 cards in a line. Each card can be randomly face up or down. You now lay – up or down – another six cards and invite him to turn over any one of the 58 cards.

The final 58 "bit" card pattern is fed into a computer and it says which card was turned over. The principle is the same as before – that is, you have "told" the computer what the original pattern of 52 cards was with just six cards. Computers do this trick all the time – it's called Hamming.

Get up to some tricks with your micro

```
5 REM(c)Computing with the Amstrad
10 DIM deck(52),decode(24)
20 REM Read the 24 ways to order 3 in
4
30 FOR k=1 TO 24
40 READ decode(k)
50 NEXT k
60 DATA 123,124,132,134,142,143
70 DATA 213,214,231,234,241,243
80 DATA 312,314,321,324,241,243
90 DATA 412,413,421,423,431,432
100 LINE INPUT"The four cards are - ";
code$
110 FOR k=1 TO 52
120 deck(k)=0
130 NEXT k
140 FOR k=2 TO 8 STEP 2
150 value=INSTR("A23456789TJQK",MID$(c
ode$,k-1,1))
160 suit=INSTR("CDHS",MID$(code$,k,1))
-1
170 deck(suit*13+value)=k/2
180 NEXT k
190 i=0
200 j=0
210 FOR k=1 TO 52
220 IF deck(k)=0 THEN 250
230 IF j>0 THEN 260
240 IF i>0 THEN j=deck(k) ELSE i=deck(
k)
250 NEXT k
260 code=100*i+10*j+deck(k)
270 FOR card=1 TO 24
280 IF code=decode(card) THEN 300
290 NEXT card
300 IF LEN(code$)>8 THEN card=card+24
310 value=-1
320 value=value+1
330 IF deck(value+1)=0 THEN card=card-
1
340 IF card>0 THEN 320
350 suit=INT(value/13)+1
360 value=value MOD 13+1
370 PRINT"The card is the ";MID$("A234
56789TJQK",value,1);" of ";MID$(".....
..Clubs  DiamondsHearts  Spades  ",su
it*8,8)
380 GOTO 100
```



Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

Program I: The CARD4 trick

Fancy another super-large mode?

THANK you for a wonderful magazine. What I have read I have thoroughly enjoyed. Keep it up.

Here is a small but useful program. It is basically a new mode, which gives you very big writing.

```
10 MODE 0
20 OUT &BC00,0:OUT &BD00,127
30 OUT &BC00,4:OUT &BD00,18
40 OUT &BC00,5:OUT &BD00,15
50 OUT &BC00,6:OUT &BD00,15
60 OUT &BC00,7:OUT &BD00,17
70 WINDOW #0,1,20,1,15
80 ORIGIN 0,160
```

All normal writing will work, and so will the graphics. Any attempt to change an ink or the border will not be recognised until after a mode change. Try using Modes 1 and 2.

To revert to the original writing, type in this:

```
10 MODE 1
20 OUT &BC00,0:OUT &BD00,63
30 OUT &BC00,4:OUT &BD00,37
40 OUT &BC00,5:OUT &BD00,0
50 OUT &BC00,6:OUT &BD00,25
60 OUT &BC00,7:OUT &BD00,30
```

I hope you will find these routines useful. They have found a place in a few of my programs. – **M. Haysman, Hainault, Essex.**

Let there be light

DURING a recent prolonged typing session at my CPC464 I became increasingly aware of the need for some form of caps lock state indication.

Would it be possible, therefore, to fit an LED, similar to the BBC Micro, to achieve this?

As an electronics engineer by profession the installation would pose no problems, but however finding where to connect the LED most certainly would.

Any comments or helpful advice in this direction would be gratefully appreciated. – **J. Findlay, North Shields, Tyne & Wear.**

■ We are unable to assist Mr Findlay with his excellent idea. Perhaps one of our readers might help him.

In search of decent descents

I AM a humble beginner (not even at novice level as yet) finding my confused way through the jungle of Basic and jargon and who has had greatness thrust on me by being made "Person in Charge of Computing" at a Birmingham school using an RML 480Z.

I have a Vic 20 plus a new Amstrad CPC464 for my own private pleasure and by sheer accident I came across your publication, which has bewildered me completely by being written in such a way that even I can understand it.

I have immediately placed a regular order with my newsagent for this most marvellous of publications (so I cannot take advantage of your reduced subscription – it's worth the extra £2 to me anyway). Now three questions:

As a teacher I want to obtain a printer that will print correctly, that is with true descenders so that my pupils will have properly printed worksheets.

Can I obtain such a machine to run with my Amstrad, and if so which one?

Is the RML 480Z the same or nearly the same as the Amstrad? I read somewhere that a program for the 380Z ran without alteration on an Amstrad.

What would I need to do to alter programs from or to Amstrad from 480Z.

Is there any way to get my Vic 1515 printer to operate on my Amstrad? – **M. Reynolds, Birmingham.**

■ Any Epson-compatible printer will print true descenders, in fact most of the more expensive printers will.

The only similiarity between the RML and the CPC464 is the Basic, and even in that there will be several variations – especially when concerned with graphics.

The Vic 1515 printer will not, we're afraid, work on the Amstrad.

Curing the jitters

RE the letter in the March issue headed "Got the jitters", from Paul O'Neil of Airdrie, the oscillation of the text on the GT64 monitor is not caused, as you suggest, by the micro. The fault lies in the monitor. I had the same trouble – indeed I noticed it on every GT64 monitor that I saw on display in various shops.

I wrote to Amstrad, who replied: "...the problem .... is in fact caused by distorted AC waveform from the mains.

"However as we have now developed a modification to overcome this problem you should return your monitor to your retailer who should contact us".

I did this as recommended and the fault has been cured. – **K. Chambers, Blackpool, Lancs.**

Overkill

I READ in January's Computing with the Amstrad *that someone did not know how to clear the keyboard buffer. The answer is:*

Call &BB03

The only problem is it also disables the Break key. So only use it when the program works. The only time you can break it is when there's an input.

I also know a little program which makes a nice display:

```
10 SPEED INK 255,255:CLS
20 CALL &BD23:CALL &BD22:
GOTO 20
```

– M.A. O'Connor, Spalding, Lincs.

■ Your call not only flashes the buffer – it does a hard reset. This is like taking a sledgehammer to crack a nut.

Your wish our command

I WAS delighted to find your magazine full of useful information.

Last month I bought the CPC464, and after loading some software I noticed a screen message: "Amsoft Presents Electro-Freddy. Loading...". Then the tape restarts and loads the main program.

My question is, how do I achieve this on a program I am writing, could you please give me the command? – **S. Joy, Hartlepool, Cleveland.**

■ There is no secret. You write a small program which contains your graphics, the last line of which should read RUN "filename". this will load your main program and run it.

One who cares

IT is becoming increasingly rare to find a computer shop that cares. One such is NSC of Manchester.

I recently took my CPC464 back to the High Street store

that I had purchased it from to repair the Space bar which had come adrift.

I was told that it would have to be sent to their workshop and could take up to two weeks to be returned.

I determined not to wait but to attempt the repair myself. However by chance I visited NSC, who not only did the repair on the spot but then insisted that there would be no charge!

I was also able to purchase several items of software that I hadn't previously been able to find locally and at very reasonable prices.

I cannot help but recommend other readers of this excellent magazine to this computer shop that not only cares but has a sense of humour as well! – **P.B.C. Gardener, Manchester.**

Point missed

UNLESS I have missed the point of Trevor Roberts' program to align decimal points (Computing with the Amstrad, March, Page 52) it would seem that it ignores the fact that there is a standard Basic command to do that very thing – PRINT USING.

I certainly would not have bought a CPC464 if it did not have this capability, as it is so difficult to display columns of numbers without it.

The following little program will achieve the same results as Roberts', but much more efficiently:

```
10 CLS
20 MODE 2
30 FOR loop= 1 TO 5
40 READ number
50 PRINT SPC(35);
60 PRINT USING "####.###";n
umber
70 NEXT loop
80 DATA 1.2,12.3,12.34,123.
45,123.456
```

The results differ from Roberts' by the inclusion of trailing zeros, which we would normally want anyway:

1.200
12.300
12.340
123.456

Many of us are still exploring the capabilities of the CPC464, so that it is inevitable that there are many gaps in our knowledge. But there is surely no excuse for publishing programs which do not exploit the near Microsoft standard Basic which is incorporated into Locomotive Basic.

I have seen many program listings which are sprinkled with % and $ signs, both of which are almost redundant in standard Basic, as one can declare integer and string variables by DEFINT and DEFSTR (incidentally, why was DEFDBL not included in Locomotive Basic?).

Most authors have realised that LET is redundant, so why do they persist in being so free with the percents and dollars?

Notwithstanding the lack of DEFDBL, the lack of Caps Lock indicator, and the lack of PRINT@ (we have to use LOCATE, then PRINT) the CPC464 is a remarkable machine for the money, and your magazine does it justice.

I have learnt many useful wrinkles (for example CALL &BC02 to return to default colours), and Mike Bibby's articles on hex, binary and machine code have been particularly informative. – **Dr M.V. Hounsome, Hale, Altrincham.**

■ I'm afraid you have missed the point. Trevor's routine is designed to accept any number and avoid rounding errors. If you replace the last piece of data in line 80 with 123.4567, say, you'll see that your routine fails in this respect.

We made an inhouse decision to keep our $ and % symbols as clearly labelling variables makes debugging easier for the less experienced.

Filling the gap

AFTER having looked through your first three issues I am most pleased that at last there is a magazine devoted to teaching the Amstrad owner how to program the CPC464 and get the most out of the machine's capabilities.

The user manual supplied is void of any real practical examples regarding many of the CPC464's functions.

I previously owned a ZX Spectrum, and although I obtained a good understanding of Basic it was only when I purchased the CPC464 that I realised the difference between Sinclair Basic and the form more widely used by other machines.

I therefore congratulate you for filling the gap of just a games magazine with a few "state of the industry" type of articles and an abundance of adverts.

Computing with the Amstrad is worth every penny and I am finding it invaluable to understand machine code (a complete mystery before), as well as your articles on sound, graphics etc.

I hope you will long continue to produce such an informative and entertaining magazine. – **P. Goodwin, Sunderland.**

Missing MERGE

I DO not suppose it took those who bought the Amstrad disc drive very long to discover that it cannot MERGE programs, although your reviewer did not mention this.

However my problem is that I thought the system was going to come with random access built in, just like the drive I had in the US for my little Radio Shack colour computer.

Unless one of your clever

Computing with the AMSTRAD Postbag

WE welcome letters from readers – about your experiences using the CPC464, about tips you would like to pass on to other users . . . and about what you would like to see in future issues.

The address to write to is:

Postbag Editor
Computing with the Amstrad
Europa House
68 Chester Road
Hazel Grove
Stockport SK7 5NY

STARFLEET

I FEEL I must congratulate you on the quality of Starfleet by Mr R. Waddilove in the March issue of Computing with the Amstrad which I have just typed in.

The game is only spoilt by the extreme stiffness of my official Amstrad joystick which almost makes me inclined to use the keyboard controls.

So far my abysmal score is 360 on screen 2.

One query though, what do the data statements in lines 1930 to 1950 do?

I have worked out how most of the rest of the program works but have obviously missed the significance of these statements. – **A.J. Billingsley, Trowbridge, Wilts.**

■ The lines you mention contain actual graphic characters with 63 subtracted from their Ascii code, so that they can be printed as a listing. This is how Roland very cleverly has printed large text on his instruction screen.

contributors can come up with a CP/M program for me I fear it will be a long, long time before I manage it myself.

I hope as time goes on that you will publish a series on CP/M along the lines of the two parallel series you are running on bits and bytes and machine code. – **A.B. Purbrick, London.**

Christian link

WE would appreciate your mentioning the newly formed Christian Micro Users Association. We hope to link together a large number of Christian micro users and also to promote the use of micros in church activities.

There is not only a great need to discover the few individuals and companies producing ''Christian'' software, but also to share the expertise and ideas of the many people who have sought to use micros in their church-related activities.

Anyone who sends a large SAE to Christian Micro Users Association, c/o 6 Walkley Street, Sheffield S6 3RG, will receive further details and a sample magazine. – **P.A. Clarke (secretary), Sheffield.**

Hard on the eyes

REGARDING the review of Level 9's Colossal Adventure and the letter from R. Pope, both published in your March 1985 edition, I have purchased both the Level 9 and the Amsoft (Abersoft) version of this adventure, and find the plot and room descriptions far superior on the cassette from Level 9.

But, like R. Pope, I find the Gothic script rather hard to read. There is a way to play this game using the Amstrad standard text though, and this is what you do:

☐ *Using LOAD'''', load the program into the CPC464. This will result in the word 'Ready' on the screen.*

☐ *ENTER the following:*

```
MEMORY &2FFF: LOAD"":CALL
&3000
```

The adventure will now load and run with the normal graphic set. – **Andy Wilkins, Westbury-On-Trym, Bristol.**

Light on windows

I ENCLOSE a short listing which may throw some light on the use of defining of windows to fill boxes with colour (S.G. Squires letter and response in the March issue of Computing with the Amstrad) *while at the same time providing a simple wrap-around slider puzzle. You use the keys Z, A, Q, W and E to move the rows and columns.* – **P.W. Hutchinson, Harlow.**

■ A very neat little program. We are sure readers will not only benefit from its programming content, but also have a lot of fun with it.

```
120 MODE 0
140 CLS
160 z=0
180 DIM p(4,3)
200 FOR y = 1 TO 3
220 FOR x = 1 TO 4
240 z=z+1
260 p(x,y)=z
280 NEXT
300 NEXT
320 FOR y = 1 TO 3
340 FOR x = 1 TO 4
360 GOSUB 1020
380 NEXT
400 NEXT
420 REM take key input
440 in$=INKEY$:IF in$="" TH
EN 440
460 in$=LOWER$(in$)
480 IF in$="q" THEN y=1:GOS
UB 640:GOSUB 1520
500 IF in$="a" THEN y=2:GOS
UB 640:GOSUB 1520
520 IF in$="z" THEN y=3:GOS
UB 640:GOSUB 1520
540 IF in$="w" THEN x=1:GOS
UB 780:GOSUB 1620:x=2:GOSUB
 780:GOSUB 1620
560 IF in$="e" THEN x=3:GOS
UB 780:GOSUB 1620:x=4:GOSUB
 780:GOSUB 1620
580 GOTO 440
600 END
620 REM re arrange top line
640 dum=p(1,y)
660 FOR x= 1 TO 3
680 p(x,y)=p(x+1,y)
700 NEXT
720 p(4,y)=dum
740 RETURN
760 REM re arrange a column
780 dum=p(x,1)
800 FOR y=1 TO 2:p(x,y)=p(x
,y+1):NEXT
820 p(x,3)=dum
840 RETURN
1000 REM locate and print w
indow
1020 WINDOW #1,4*x-1,4*x+2,
5*y-2,5*y+2
1040 PAPER #1,p(x,y)
1060 PEN #1,13-p(x,y)
1080 CLS #1
1100 LOCATE #1,1,1
1120 PRINT #1,CHR$(64+p(x,y
));
1140 LOCATE #1,2,2
1160 PRINT #1,CHR$(64+p(x,y
));
1220 RETURN
1500 REM alter a line
1520 FOR x= 1 TO 4
1540 GOSUB 1020
1560 NEXT
1580 RETURN
1600 REM alter a column
1620 FOR y= 1 TO 3
1640 GOSUB 1020
1660 NEXT
1680 RETURN
```

Riddle of code 25

COULD anyone explain how to use control code 25 decimal on the Amstrad?

In my machine code program I wanted to redefine a character for a particular use. Using control code 25 dec, followed by the 9 bytes as specified in the manual just doesn't work.

These were sent out to BB5A-TXT OUTPUT, which should obey control codes.

The only way that I have found so far of redefining a character is shown in this short assembler listing:

```
LD DE,255
LD HL,MATRIX
PUSH HL
CALL BBAB
POP HL
LD A,255
CALL BBA8
```

*This does rather limit you as to the choice of which character to redefine. Otherwise you have to set up a matrix 256 – first character *8 long.* – **R.G. Bennett, Rotherham, S. Yorks.**

■ The subroutine TXT OUTPUT (&BB5A) is supposed to accept all control characters from 0 to &1F, but for some unknown reason code 25 fails.

Can anyone out there help us find out why the code is being incorrectly executed?

Video hook-up

I WRITE with reference to the letter from J.D. Smith of Fareham, Hants (March 1985 issue) concerning colour monitors and video recorders.

You may be interested to know that I am using a Fidelity CTM1400 monitor/TV successfully with the CPC464 using the RGB output direct to the CTM1400 via the latter's SCART input socket.

In addition the CTM1400 can be used with a video

recorder using the recorder's video output socket and audio output.

As the SCART input is a 21 pin device (plug readily available), both the CPC464 and the video recorder output with audio can be wired via the same plug.

This assumes of course that the CPC464 is not in use at the same time as the video recorder, otherwise separate leads can be made up.

Picture quality from both video recorder and CPC464 is outstanding.

I wish continued success to your magazine. Beats its competitors hands down, including the 'Official Mag!' – **J.E. Vivian, Cheltenham, Glos.**

Editor markers

MAY I suggest a couple of additional lines to your February Text Editor to provide markers at the top of the screen to show where the currently set tabs are:

```
2025 PRINT #4,SPC(column%(i
)-column%(i-1)-1);CHR$(211)
;
2135 PRINT #4,SPC(71-column
%(4));
435 LOCATE #4,5,1:FOR i=1 T
O 4:PRINT#4,SPC(column%(i)-
column%(i-1)-1);CHR$(211);:
NEXT:PRINT #4,SPC(71-column
%(4));
```

And what if you realise that you've started printing and there's a mistake in the text? You don't want to wait for the whole thing to print out, do you? So put in these two lines:

```
2335 LOCATE #1,23,3:PRINT#8
,"Press SPACE BAR to interr
upt"
2355 IF INKEY(47)>-1 THEN R
ETURN
```

I have found that these help enormously, and perhaps they will help other readers. – **Jim Barton, Beverley, North Humberside.**

Relocating screen dump

I READ with interest the article by Roland Waddilove on a screen dump utility (Computing with the Amstrad, March 1985) and being the owner of an Epson RX80FT promptly set about entering the listing.

When finished I followed the instructions on running the program only to have an error report – "Memory full". Not being what I would call a proficient programmer, I set out to try to discover the cause of the offending error.

After making absolutely sure that the listing had been typed in correctly it occurred to me that perhaps it was not written with the disc drive in mind.

So being new to DOS I read through the User Guide to see where in memory the system lives, and there on page F 3.2 is the warning DDI-1 interface reduces memory by 1280 bytes. So what do I do now?

OK, let's add this subtract that, mess about with all these funny letters that some people say they count with and 2k cups of tea later this is what I find. Line 120 Memory Himem etc.

And, would you believe I get a screen dump. Now for a guy who doesn't really know what he's at with machine code this was real progress.

However after more key bashing I found that the whole thing can be simplified to:- 120 sum=0. Yes, that's right, no memory setting and it still works.

Still any other disc user that found the same problem will now be able to make the necessary mod and dump to printer without the need to unplug DDI-1.

By the way, you can actually set memory at anything reasonable. Try &A5B0 as an example. But then of course FRE(0) will show a vast reduction in memory pool, so the program you wish to dump will be limited.

Try the above and then do PRINT FRE(0) and see how much more memory there is. – **L.J.Batty, Castleford, West Yorkshire.**

■ Although you appear to have found a solution to the problem, it is in fact an unsatisfactory one.

The screen dump loads to the same area of memory as the disc drive and will disable the drives. Here is a better solution for anyone with access to an assembler.

The screen dump will work if it is relocated. The only way is to type in the assembly listing using an assembler and set ORG to about &A000 and reset himem. This should be well clear of anything.

The variable "colour" is the colour to be ignored by the dump – that is, the paper. Set this to what ever you want (you can always POKE start +&4F, colour afterwards).

```
POKE start + &4F, colour
```

The assembler listing is correct except that the bottom three instructions in the last column should be at the top – they must have been pasted in the wrong place.

I hope they didn't cause too much confusion.

Roland Waddilove

Disc drive hang-ups

THE word processing program by Roland Waddilove in the February issue of Computing with the Amstrad has proved extremely useful.

However when I added a disc drive to my Amstrad it did not perform as expected, giving me unexpected hang-ups when the load, save or end program options were selected.

I assume this is because of memory location problems associated with the disc drive, and the disc requirements for loading and saving.

I should be most grateful to hear if there is a modification to the program which will solve these problems. – **B.M. Howe, Bristol.**

■ Here are a few notes on how to alter TextEd so that it will run with the disc drive.

The machine code must be located lower down in the memory for the program to run properly. The code should be OK at about &A000, so go through the program carefully and change all CALL &ABXX, to CALL &A0XX.

Change the following lines:

```
  260 change POKE &AB00 to
POKE &A000
  280 change last hex no.
from AB to A0
  290 change last hex no.
from AB to A0
  300 change 9th hex no.
from 6A to 5F
  370 change   last =
40959-length
 1720 change to OPENOUT
name$+"1"
 1840 change to OPENIN
name$+"1"
```

"Ebagum"

WHEN INPUTting a string, if a comma or any non-letter is present I am often instructed to "?Redo from Start".

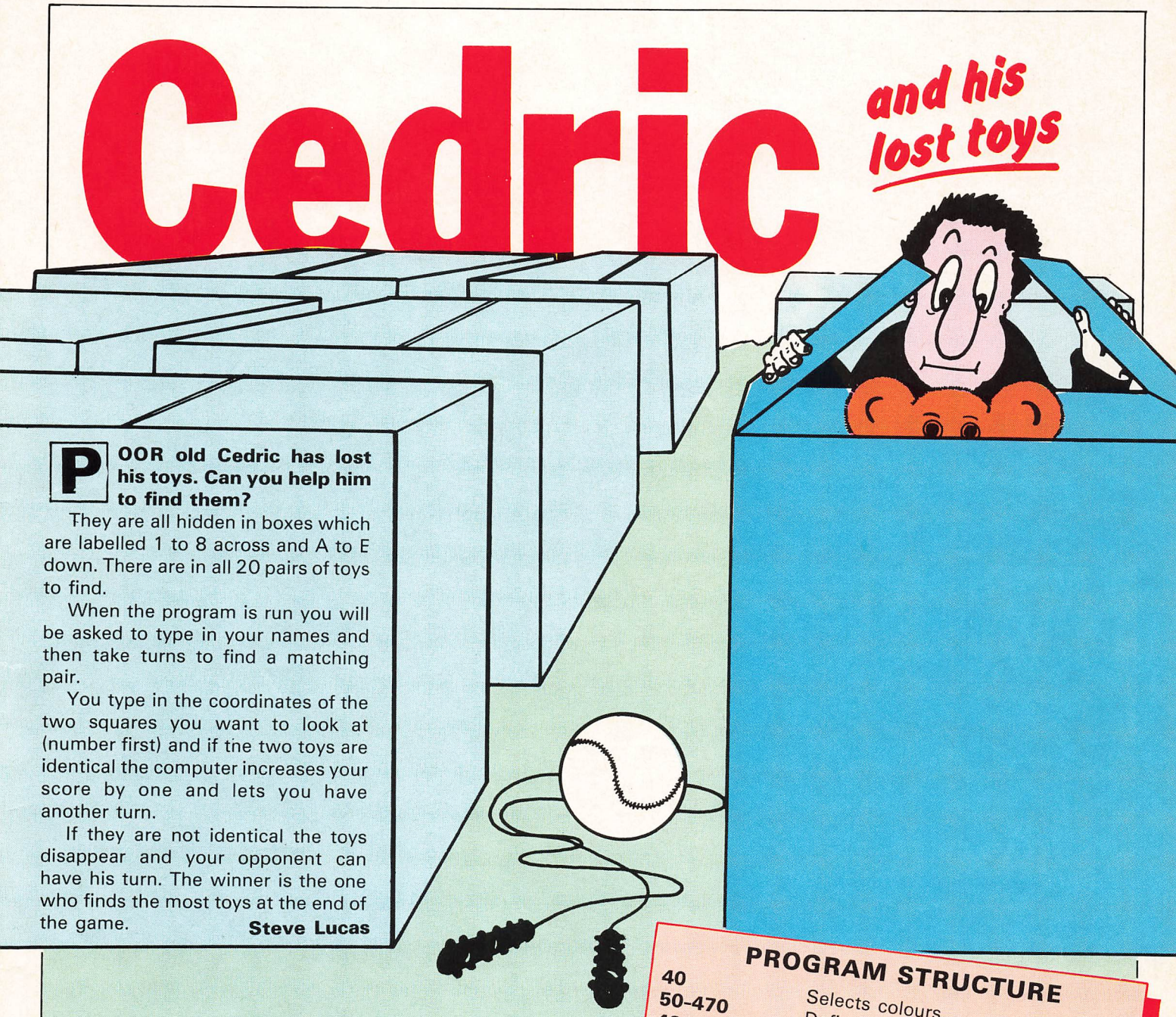
Try entering "EBAGUM, MUGABE" on the Palindrome Tester, in the April issue, and you'll see what I mean.

Finally, congratulations to Mike Bibby for his excellent machine code tutor series, but could he please give a ready-reference chart of opcodes and firmware routines. – **William Redgrave, Maidenhead, Berks.**

■ INPUT uses commas to separate different items, so EBAGUM, MUGABE is united as two inputs – see this month's beginners' article on page 16.

As for the tables, this month's machine code article should provide what you need. Turn to page 26.

Cedric and his lost toys



POOR old Cedric has lost his toys. Can you help him to find them?

They are all hidden in boxes which are labelled 1 to 8 across and A to E down. There are in all 20 pairs of toys to find.

When the program is run you will be asked to type in your names and then take turns to find a matching pair.

You type in the coordinates of the two squares you want to look at (number first) and if the two toys are identical the computer increases your score by one and lets you have another turn.

If they are not identical the toys disappear and your opponent can have his turn. The winner is the one who finds the most toys at the end of the game.

Steve Lucas

VARIABLES

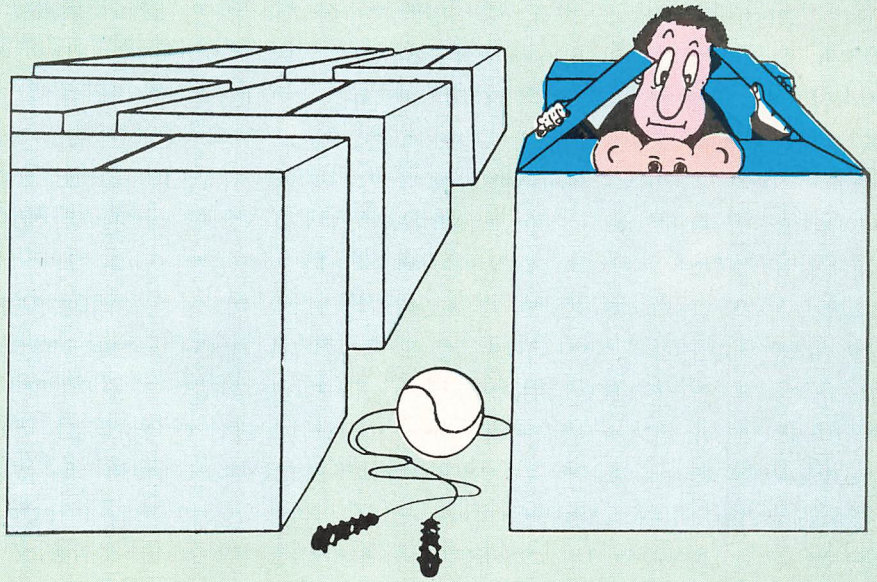
name1$,name2$	Names of the players.
s1%,s2%	Scores of the two players.
j%	Number of pairs of toys found.
z%	Used as a flag.
aa$-at$	Graphics for toys.
ob%(x,y)	Array holds the toys.
a,b	Random numbers to hide toys.
x,y	Coordinates for graphics.
t$,s$,t,s	Input squares.
fa%	Check for match.
qa,qb,pa,pb	Coordinates of toys displayed.

PROGRAM STRUCTURE

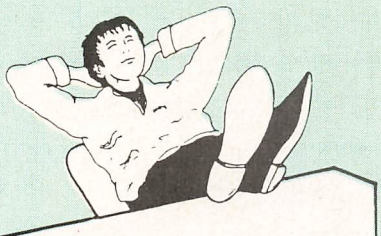
40	Selects colours.
50-470	Define characters.
480-690	Define graphics.
700	Randomises program.
710	Titles.
720-790	Hide toys in array.
800-820	Wait to start game.
840-850	Input names.
860-1010	Draw board.
1030	Labels axes.
1050-1070	Main control loop.
1080-1150	End game and display scores.
1160-1410	Enter coordinates and display toys.
1420-1440	End of turn.
1450-1490	Erase graphics.
1500-1540	Move to correct coordinates.
1550-1760	Examine array and print toys.
1770-1870	Titles and instructions.
1890-1930	Tune.
1940-1960	Set scores.

```
10 REM ** Cedric and the lost toys **
20 REM ** by Steve Lucas **
30 REM (c) Computing with the Amstrad
40 MODE 1:INK 0,1:INK 1,24:INK 2,20:I
NK 3,6
50 REM ** define the characters **
60 SYMBOL AFTER 200
70 SYMBOL 215,128,128,255,3,255,128,1
28,128
80 SYMBOL 216,1,1,255,64,255,1,1,1
90 SYMBOL 217,156,156,144,144,144,255
,128
100 SYMBOL 218,0,0,0,0,0,0,128,128
110 SYMBOL 219,7,3,1,1,1,1,3,7
120 SYMBOL 220,240,224,192,192,192,19
2,224,240
130 SYMBOL 221,15,24,63,5,13,9,18,20
140 SYMBOL 222,248,140,254,40,236,36,
18,10
150 SYMBOL 223,14,31,55,127,63,3,14,0
160 SYMBOL 224,0,8,152,248,152,24,8,0
170 SYMBOL 225,1,1,1,1,1,1,39,31
180 SYMBOL 226,192,192,192,192,196,19
6,245,254
190 SYMBOL 227,15,63,255,25,31,25,31,
31
200 SYMBOL 228,240,252,255,216,248,24
8,216,216
210 SYMBOL 229,0,96,112,88,204,252,0,
192
220 SYMBOL 230,103,159,89,185,95,187,
44,71
230 SYMBOL 231,230,249,154,157,250,22
1,52,226
240 SYMBOL 232,192,128,156,191,255,25
5,65,113
250 SYMBOL 233,1,255,255,255,255,0,0,
0
260 SYMBOL 234,24,24,48,255,255,20,20
,60
270 SYMBOL 235,0,2,2,250,254,40,40,12
0
280 SYMBOL 236,0,0,15,11,15,255,127,6
3
290 SYMBOL 237,128,128,240,240,240,25
5,255,255
300 SYMBOL 238,0,254,222,142,222,254,
254,12
310 SYMBOL 239,0,15,9,9,127,127,127,4
8
320 SYMBOL 240,0,0,255,253,253,220,20
,60
330 SYMBOL 241,136,112,127,127,127,14
,10,30
340 SYMBOL 242,192,224,224,224,248,20
4,14,174
350 SYMBOL 243,1,3,3,3,15,25,56,58
360 SYMBOL 244,255,133,255,5,255,255,
12,12
370 SYMBOL 245,255,136,255,136,255,25
5,24,24
380 SYMBOL 246,0,160,224,160,235,255,
53,63
390 SYMBOL 247,0,2,3,2,127,255,170,25
4
400 SYMBOL 248,240,156,248,72,72,16,1
44,224
410 SYMBOL 249,7,28,15,9,9,4,4,3
420 SYMBOL 250,0,0,192,48,252,255,24,
24
430 SYMBOL 251,0,0,15,24,127,225,24,2
4
440 SYMBOL 252,0,0,0,8,232,252,8,8
450 SYMBOL 253,0,128,143,241,255,127,
3,2
460 SYMBOL 254,128,224,0,0,255,254,25
2,248
470 SYMBOL 255,1,1,1,1,255,127,63,31
480 REM ** define the graphics for to
ys **
490 aa$=CHR$(255)+CHR$(254)
500 ab$=CHR$(253)+CHR$(252)
510 ac$=CHR$(251)+CHR$(250)
520 ad$=CHR$(245)+CHR$(244)
530 ae$=CHR$(243)+CHR$(242)
540 af$=CHR$(249)+CHR$(248)
550 ag$=CHR$(247)+CHR$(246)
560 ah$=CHR$(241)+CHR$(240)
570 ai$=CHR$(239)+CHR$(238)
580 aj$=CHR$(236)+CHR$(237)
590 ak$=CHR$(234)+CHR$(235)
600 al$=CHR$(232)+CHR$(229)
610 am$=CHR$(230)+CHR$(231)
620 an$=CHR$(227)+CHR$(228)
630 ao$=CHR$(225)+CHR$(226)
640 ap$=CHR$(223)+CHR$(224)
650 a1$="  "
660 aq$=CHR$(221)+CHR$(222)
670 ar$=CHR$(219)+CHR$(220)
680 as$=CHR$(217)+CHR$(218)
690 at$=CHR$(215)+CHR$(216)
700 RANDOMIZE TIME
710 GOSUB 1780:REM ** instructions **
720 REM ** HIDE OBJECTS IN THE ARRAY
**
730 DIM ob%(10,10)
740 z%=1
750 FOR y=1 TO 8:FOR x=1 TO 5
760 a=INT(RND(1)*5)+1:b=INT(RND(1)*8)
+1
770 IF ob%(a,b)<>0 THEN 760
780 ob%(a,b)=z%:z%=z%+1:IF z%>20 THEN
 z%=1
790 NEXT x,y
800 REM ** wait to start game **
810 PRINT:PRINT"   Press the <Space B
ar> to start."
820 a$=INKEY$:IF a$<>" "THEN 820
830 REM ** input players names **
840 CLS:PEN 1:INPUT"Player 1 please e
nter your name          ";name1$:PRINT
 CHR$(7)
850 INPUT"Player 2 please enter your
name          ";name2$:PRINT CHR$(7)
860 REM ** draw the board **
870 CLS
880 ORIGIN 1,100
890 FOR x=0 TO 8
900 MOVE x*64,1: DRAWR 0,350,1
910 NEXT x
920 WINDOW #1,1,35,20,25:PAPER #1,2:C
LS#1
930 MOVE 0,299:DRAWR 550,0,1
940 TAG
950 MOVE 0,260:DRAWR 550,0,1
960 MOVE 550,0:DRAWR 0,299,1
970 FOR y=0 TO 4
980 MOVE 0,y*52: DRAWR 550,0,1
990 MOVER -30,30:PRINT CHR$(69-y);
1000 NEXT
1010 TAGOFF
1020 REM ** label axis **
1030 PEN 2:FOR x= 0 TO 7: LOCATE x*4+
2,2:PRINT x+1;:NEXT x
1040 REM ** main control loop...repea
ted until all toys have been found **
1050 WHILE j%<20
1060 GOSUB 1170
1070 WEND
1080 REM ** end of game.. display sco
res **
1090 CLS#1:PEN#1,0:PRINT#1,"Well Done
 you found all the objects"
1100 PEN #1,3:PRINT#1,name1$;" found
";s1%;" toys"
1110 PRINT#1,name2$;" found ";s2%;" t
oys"
1120 PRINT#1:PRINT#1,"  Press <Space
Bar> to play again"
1130 aa$=INKEY$:IF aa$<>" " THEN 1130
1140 RUN
1150 END
1160 REM ** guess coordinates **
1170 CLS#1:PEN #1,0:PRINT#1,SPC(5);"C
```

```
edric and the lost toys":PEN #1,3
1180 PEN #1,0:IF pl%=0 THEN PRINT#1,n
ame1$;SPC(5);"score ";s1% ELSE PRINT#
1,name2$;SPC(5);"score ";s2%
1190 PRINT#1
1200 PRINT#1,"Enter your first guess
  ";:PEN #1,3
1210 s$=INKEY$:IF s$="" THEN 1210
1220 s$=UPPER$(s$):IF ASC(s$)>56 OR A
SC(s$)<49 THEN 1210
1230 PRINT#1,s$;" ";
1240 t$=INKEY$:IF t$="" THEN 1240
1250 t$=UPPER$(t$):IF ASC(t$)>69 OR A
SC(t$)<65 THEN 1240
1260 PRINT#1,t$
1270 GOSUB 1510
1280 IF fa%=1 THEN CLS#1:GOTO 1170:RE
M ** check if already guessed **
1290 p=ob%(t,s):pb=t:pa=s
1300 PEN #1,0:PRINT#1,"Enter your sec
ond guess  ";:PEN #1,3
1310 s$=INKEY$:IF s$="" THEN 1310
1320 s$=UPPER$(s$):IF ASC(s$)>56 OR A
SC(s$)<49 THEN 1310
1330 PRINT#1,s$;" ";
1340 t$=INKEY$:IF t$="" THEN 1340
1350 t$=UPPER$(t$):IF ASC(t$)>69 OR A
SC(t$)<65 THEN 1340
1360 PRINT#1,t$
1370 GOSUB 1510
1380  IF fa%=1 THEN CLS#1:GOTO 1300:R
EM ** check if already guessed **
1390 q=ob%(t,s):qb=t:qa=s
1400 IF qa=pa AND qb=pb THEN PRINT CH
R$(7):GOTO 1300
1410 IF p=q THEN CLS#1:j%=j%+1:ob%(qb
,qa)=0:ob%(pb,pa)=0:GOSUB 1890:GOSUB
1950:RETURN
1420 REM ** end of turn **
1430 CLS#1:LOCATE #1,1,2:PRINT#1,"Pre
ss the <Space Bar> to continue."
1440 a$=INKEY$:IF a$<>" " THEN 1440
1450 REM ** erase graphics **
1460 LOCATE 2+(pa-1)*4,5+(pb-1)*3:PRI
NT"  ";
1470 LOCATE 2+(qa-1)*4,5+(qb-1)*3:PRI
NT"  ";
1480 pl%=pl%+1:IF pl%>1 THEN pl%=0
1490 RETURN
1500 REM ** print graphics at correct
 coordinates **
1510 t=ASC(t$)-64:s=ASC(s$)-48
1520 fa%=0
1530 LOCATE 2+(s-1)*4,5+(t-1)*3
1540 IF ob%(t,s)=0 THEN fa%=1:PRINT C
```



```
HR$(7):RETURN
1550 REM ** check contents of array t
o find out which toy is there **
1560 IF ob%(t,s)=1 THEN PRINT aa$
1570 IF ob%(t,s)=2 THEN PRINT ab$
1580 IF ob%(t,s)=3 THEN PRINT ac$
1590 IF ob%(t,s)=4 THEN PRINT ad$
1600 IF ob%(t,s)=5 THEN PRINT ae$
1610 IF ob%(t,s)=6 THEN PRINT af$
1620 IF ob%(t,s)=7 THEN PRINT ag$
1630 IF ob%(t,s)=8 THEN PRINT ah$
1640 IF ob%(t,s)=9 THEN PRINT ai$
1650 IF ob%(t,s)=10 THEN PRINT aj$
1660 IF ob%(t,s)=11 THEN PRINT ak$
1670 IF ob%(t,s)=12 THEN PRINT al$
1680 IF ob%(t,s)=13 THEN PRINT am$
1690 IF ob%(t,s)=14 THEN PRINT an$
1700 IF ob%(t,s)=15 THEN PRINT ao$
1710 IF ob%(t,s)=16 THEN PRINT ap$
1720 IF ob%(t,s)=17 THEN PRINT aq$
1730 IF ob%(t,s)=18 THEN PRINT ar$
1740 IF ob%(t,s)=19 THEN PRINT as$
1750 IF ob%(t,s)=20 THEN PRINT at$
1760 RETURN
1770 REM ** titles and instructions *
*
1780 CLS:WINDOW #2,4,36,1,3:PEN #2,3:
PAPER #2,1:CLS#2
1790 LOCATE #2,6,2:PRINT#2,"Cedric an
d the lost toys"
1800 LOCATE 6,6:PRINT"A game for the
Amstrad CPC464"
1810 LOCATE 11,8:PEN 2:PRINT"by  Stev
e W. Lucas"
1820 LOCATE 2,10:PEN 1:PRINT"Cedric h
as lost his toys and doesn't  know
where to find them."
1830 PRINT"Can you help him? The toys
 are hidden onthe board, which is lab
elled A to E downand 1 to 8 across."
1840 PRINT"You can look at what is in
 a square by  typing in the coordinat
es (number first)";
1850 PRINT"If the two toys are the sa
me, you will  score one and the toys
will remain on  the screen. If they
are not, they will  disappear."
1860 GOSUB 1890
1870 RETURN
1880 REM ** tune for finding matching
 toys **
1890 RESTORE:FOR x=1 TO 6
1900 READ d:SOUND 1,d,10
1910 NEXT
1920 DATA 478,478,426,478,358,426
1930 RETURN
1940 REM ** set the scores if match f
ound **
1950 IF pl%=0 THEN s1%=s1%+1 ELSE s2%
=s2%+1
1960 RETURN
```



Give your fingers a rest . . .

All the listings from this month's issue are available on cassette.

See our special offer on Page 77.

CASTLE BLACKSTAR

"A cut above the usual standards, could well become a classic."
POPULAR COMPUTING WEEKLY

"Highly recommended!"
HOME COMPUTER WEEKLY

"Where this program really scores is in the description of its locations and the large (200 word plus) vocabulary."
MICRO ADVENTURER

"Great game and I look forward to Artemis Quest 2".
CRASH

"An excellent, absorbing, detailed and tough adventure."
COMPUTER AND VIDEO GAMES

£6.95 inc. P. & P. CPC464

CDS SOFTWARE LTD.
Silver House, Silver Street,
Doncaster DN1 1HL
Tel: (0302) 21134

UTILITY OF THE YEAR
CRASH Readers Awards 1984
Winner
THE YEAR

The Quill

Adventure Writing System

VOTED

UTILITY OF THE YEAR

BY CRASH MICRO & POPULAR COMPUTING WEEKLY

NOW AVAILABLE FOR

THE AMSTRAD

C.P.C.464

✂----------

Please send me order form and details of
The Quill for The Amstrad C.P.C.
I Enclose an S.A.E.

Name .....

Address .....

.....

.....

Cassette £16.95.

Disk T.B.A.

Send to;-
GILSOFT,
30 Hawthorn Road,
Barry, S. Glamorgan.
0446 - 732765.

Credit Card Order Line Staffed 24 Hours 0222 - 41361 Ext 430

Now you've started computing with the Amstrad you'll want to read EVERY issue!

FREE! Send in your subscription on the form below and we'll send you this month's cassette, worth £3.75, entirely free of charge!

Give your fingers a rest

All the program listings from each month's issue are available for only ***£3.75*** *on cassette and* ***£6.75*** *on disc.*

JANUARY: ***Smiley:*** *Avoid the Grumpies with Smiley round the labyrinth.* ***Code:*** *Be a mastermind and play this intriguing logic game.* ***Binary:*** *Baffled by binary bits? Let our utility help you out.* ***Dancer:*** *Simple but fun. This is a lovely mover.* ***Trapper:*** *Pen the monsters of the maze.* ***Scroller:*** *A slick sideways scrolling routine.* ***Letter Litter:*** *Learn the keyboard with this entertaining educational game.*

FEBRUARY: ***Digger:*** *Search for crystals and avoid the aliens.* ***Simon:*** *Our Amstrad version of this simple children's game.* ***Kingdom of Craal:*** *A devious fantasy full of surprises.* ***Text Editor:*** *A powerful, yet simple word processor.* ***Reaction Timer:*** *Test out your reactions on this simple little program.* ***Origins:*** *The effects of changing the graphics origin produce some dramatic effects.*

MARCH: ***Starfleet:*** *Can you qualify as a starfleet pilot?* ***Swamp:*** *Join the desperate battle against the giant frogs.* ***Parachute:*** *Guide your parachute to earth in this educational program.* ***Dump:*** *An invaluable screen dump utility for Modes 0 and 1.* ***Hexer:*** *A hexadecimal loader.* ***PLUS*** *an extra full-length game* ***3D Four in a Row*** *– an enthralling game of luck, skill and strategy.*

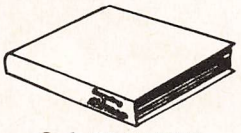
APRIL: ***Mad Adder:*** *Appease our aggravated adder's appetite.* ***Egg Blitz:*** *Eggs, boy scouts and a stunt pilot make up the fun.* ***Simple Sprites:*** *Create sprite-like effects for your programs.* ***Caesar Cipher:*** *Learn how to crack the code.* ***Pilot:*** *An easy to learn language, ideal for computer aided learning.* ***Spelling:*** *Learn to spell with this educational game.* ***Aleatoire:*** *Find out the odd ways mariners make decisions.* ***Palindrome Tester:*** *Slick string handling techniques.*

MAY: ***Tronn Cycles:*** *Enter the Maze of Death and pit your wits in this two player arcade classic.* ***Castle of Fear:*** *Fantasy is the order of the day in this baffling concoction by the devious A Team.* ***Mouse:*** *Can you trap and pounce on the furry fellow in this simple but addictive game?* ***Cedric:*** *Have fun with shape recognition in this novel interpretation of pelmanism.* ***Character Maker:*** *Define your own characters with our superb utility.* ***RSX:*** *Provide and implement new commands for your micro.* ***Cards:*** *Play the conjurer and baffle your friends with an apparently simple computer card trick.* ***Trees:*** *A clever routine to create 'trees' from Trevor Roberts' Analysis.*

New! We are pleased to be able to offer you annual subscription for the monthly listings cassette for only £40, and on 3" disc for only £70.

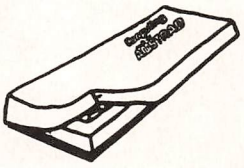
How to keep your collection complete

Bound in rich burgandy pvc and bearing the Computing with the Amstrad logo, this handsome binder will hold a year's supply of the magazines firmly secured in place with metal rods.



Only £3.95 (UK)

Protect your keyboard with our luxury dust cover made of soft, pliable, clear and water-resistant vinyl, bound with strong cotton and decorated with the magazine's logo.



Dust Cover Only £3.95

ORDER FORM

All prices include postage, packing and VAT and are valid to June 29

Please enter your requirements by ticking boxes

			£	p
Annual subscription	UK £12	6001		
	EIRE £13 (IR £16)	6002		
	Overseas (Surface) £20	6003		
	Overseas (Airmail) £40	6004		
Please send me my free April issue listings cassette				
Commence with _____ issue			TOTAL	
Back issues £1.25 UK £1.50 Overseas (Surface)	January	6006		
	February	6007		
	March	6008		
	April	6009		
			TOTAL	
Tapes £3.75 (UK & Overseas)	January (Smiley)	6020		
	February (Digger)	6021		
	March (Starfleet)	6022		
	April (Mad Adder)	6023		
	May (Tronn Cycles)	6024		
			TOTAL	
3" Discs £6.75 (UK & Overseas)	January (Smiley)	6060		
	February (Digger)	6061		
	March (Starfleet)	6062		
	April (Mad Adder)	6063		
	May (Tronn Cycles)	6064		
			TOTAL	
Cassette tape and disc annual subscription Tape £40 (UK & Overseas)		6005		
3" Disc £70 (UK & Overseas)		6059		
Commence with _____ tape/disc (state month)			TOTAL	
Dust Cover £3.95 (UK & Overseas)		6037		
			TOTAL	
Binder £3.95 UK £5.00 Overseas		6038		
			TOTAL	
Payment: please indicate method (✓)			**TOTAL**	

☐ Access/Mastercharge/Eurocard

Card No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Barclaycard/Visa

Card No. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____ Signed _____

Address _____

Send to:
Computing with the Amstrad, **FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.**
(No stamp needed if posted in UK) *Please allow 28 days for delivery*

YOU CAN ALSO ORDER BY PHONE:

All other enquiries 061-480 0171 **Orders (24 hours) 061-480 0173**

Don't forget to quote your credit card number and full address

CUT LOADING TIME DRAMATICALLY AND LIST YOUR WELCOME TAPE WITH OUR WELL KNOWN "SYCLONE" PROGRAM

* * * *

TRANSFER YOUR PROGRAMS ONTO DISC WITH OUR EXCITING NEW PROGRAM "TRANSMAT"

NOW AVAILABLE SCRIPTOR FOR THE DMP 1 PRINTER OWNER

* * * *

Always the first and the best software, offering more features and better value for money than other similar programs available. We also offer a fast reliable and friendly mail-order service. Look at just some of the features our programs offer.

RSX. SYCLONE

Copy and or convert your programs to load in up to 4 times faster.
Features include: + Commands available from Basic + Choice of 4 loading speeds, 1000 to 4000 baud + Copies all tested software + Comprehensive header reader + Load and list protected basic programs.

Cassette £6.95 inc P&P

TRANSMAT

Transfer your software onto the Amstrad Disc System (DDI-1)
Features include: + Faithfully transfer all programs + Add relocater if necessary + Auto or non-auto modes + Erase or rename programs + Comprehensive header reader.

Cassette £7.95 inc P&P

ZEDIS II

A comprehensive machine code editor and disassembler.
Features include: + Continual menu display + Break point insertion + Register inspection + High speed hex.code/string search + Hex.code/string input.
Instructions included to disassemble the ROMs.

Cassette £6.95 inc P&P
Disc £10.95 inc P&P

PRINTER PAC 1

A Printer enhancement program for the DMP1 and Epson compatible printers such as the Epson RX80 and Shinwa CPA80.
Features include: + Screen dump in all modes + 2 sizes of dump for Epson compatible printers + Text dump in all modes + 3 new type styles for the DMP1 + Abbreviated codes to printer.

Cassette £5.95 inc P&P
Disc £9.95 inc P&P

SCRIPTOR

For the DMP1 Printer. This marvellous program allows 6 type styles to be used on the DMP1.
Including realwriting, italics etc. on sets of your own design. It also overcomes that lower case descender problem brilliantly. The program pack only £6.95 inc P&P or £10.95 on disc. S.A.E. for details.

ALL DISC BASED TITLES HAVE FREE DISC SPACE AVAILABLE TO USER

*** SPECIAL OFFER ***
Worth £3.95
Buy more than one title and get a cassette containing a real time Digital Alarm Clock FREE (while stocks last)

PRIDE UTILITIES LTD (CWA)
7 Chalton Heights, Chalton, Luton, Beds. LU4 9UF.

Europe – ADD £1 per title

Rest of World – ADD £1.50 per title

Hobbit	£12.85	Jet Set Willy	£7.50
Sorcery	£7.95	Fighter Pilot	£7.95
Dark Star	£6.95	Technician Ted	£6.95
Redcoats	£5.95	Ghouls	£6.00
S. Davis Snooker	£6.95	All Level 9	£8.35
Emerald Isle	£6.95	All Interceptor	£5.25
All Anirog	£5.95	All Amsoft	£7.50
Bridge Player	£7.50	Defend or Die	£6.75
Football Manager	£6.95	Mission 1 Volcano	£7.95
Basic Pt. 1	£17.00	Basic Pt. 2	£17.00
M/C Tutor	£13.00	American Football	£8.75
Mutant Monty	£5.95	Stockmarket	£7.50
Pyjamarama	£7.50	Erbert	£5.25
Fantasia Diamond	£6.95	The Pyramid	£6.50
Backpackers Guide	£6.50	Amsword	£17.00
Devpac	£20.50	All Bourne	£7.50
Sir Lancelot	£5.95	Heathrow Int ATC	£6.95
Screen Designer	£10.95	The Tripods	£10.50

Cheques/P.O. to: *Write or phone for free catalogue*

MICRO COMPUTER WORLD
1 LANE CLOSE, LONDON NW2 6QZ. Tel: 01-452 0893
Beware of cheap bootleg software – all our titles are originals

SHEKHANA COMPUTER SERVICES

	RRP	Our Price		RRP	Our Price
Air Traffic Control	7.95	6.75	Screen Designer	14.95	12.95
American Football	9.99	8.25	Star Avenger	6.95	5.95
Sorcery	8.95	7.90	Star Commando	8.95	7.75
Chess (Mikgn)	6.95	5.75	Steve Davis Snooker	7.95	6.50
Blagger	8.95	7.90	Fruity Frank (Kuma)	6.95	5.75
Dark Star	7.95	6.75	Technician Ted	7.95	6.90
Gillians Gold	6.95	6.25	Test Match	6.95	5.90
Flight Path 737	6.95	5.90	Forest at Worlds End	6.00	4.99
Football Manager	7.95	6.75	Message from Andromeda	6.00	4.99
Gems of Stradus	7.95	6.75	Jewels of Babylon	6.00	4.99
Fighter Pilot	8.95	7.90	Heroes of Karn	6.00	4.99
Jet Set Willy	8.95	7.50	Chopper Squad	6.00	4.99
Manic Minor	8.95	7.50	Munch It	6.95	5.90
Mission 1 Volcano	8.95	7.90	Zen – Assembler (Kuma)	19.95	16.75
All Level 9 Games	9.95	8.25	Guide to Basic Part 1	19.95	15.00
Pyjamarama	6.95	5.75	Azimuth 3000	8.99	7.90

Cheques/PO's to:
SHEKHANA COMPUTER SERVICES
653 Green Lanes N8 0QY London
(Mail Order Address only)
Tel: 01-800 3156 (S.A.E. for List)
– or visit our shop at Marbles Shopping Centre, Unit 11, 527-531 Oxford Street, W1R 1DD.
Open 7 days a week from 10am – 19.00pm (opps. Marble Arch Tube Station)
Above discounts applicable only on production of this advert.

SIMPLE ACCOUNTS for the AMSTRAD CPC 464

- FULL ANALYSIS OF INCOME & EXPENDITURE
- POWERFUL ENTRY SEARCH ROUTINES
- CONSTANT UPDATE OF CREDITORS AND DEBTORS
- SPECIAL IN-DEPTH ANALYSIS OPTION
- VAT REPORT
- DATE SORT
- MONTHLY AND YEAR TO DATE REPORTS
- FULLY DOCUMENTED
- EASY TO USE

ONLY £29.95 including VAT and carriage

Full specification manual and worked examples available on request.

CORNIX SOFTWARE LTD
16 Kneesworth Street, Royston, Herts. SG8 5AA.
Tel: Royston (0763) 46065
Specialists in business and financial software for the serious user.

SUNARO Software

BEST FOR AMSTRAD SOFTWARE
Order any two titles deduct £1 extra

SURVIVOR	6.25	MUTANT MONTY	7.85	ROLAND ON THE ROPES	7.85
SOFTWARE STAR	6.95	CENTRE COURT	7.85	ROLAND AHOY	7.85
DARK STAR	6.95	DEFEND OR DIE	6.95	ROLAND IN THE CAVE	7.85
FIGHTER PILOT	7.85	HOME RUNNER	7.85	TECHNICIAN TED	7.85
HEATHROW ATC	6.95	HAUNTED HEDGES	7.85	MUSIC COMPOSER	8.75
CHOPPER SQUAD	5.25	THE HOBBIT	13.25	ERIK THE VIKING	8.75
SPECIAL OPERATIONS	6.25	JEWELS OF BABYLON	5.25	TASWORD 464	17.95
BATTLE FOR MIDWAY	8.75	MESSAGE ANDROMEDA	5.25	TASPRINT 464	8.75
TRIPODS	10.25	FOREST AT WORLDS END	5.25	CONCISE BASIC	10.95
SORCERY	7.85	STEVE DAVIS SNOOKER	6.95	CONCISE FIRMWARE	17.95
MACHINE CODE TUTOR	13.25	FLIGHT PATH 737	6.25	SNOOKER	7.85
EMERALD ISLE	7.85	FRUITY FRANK	6.25	GRAND PRIX DRIVER	7.85
ER BERT	5.25	MANIC MINER	6.25	FANTASIA DIAMOND	6.95

New titles available immediately upon release
Cheque/PO to:
SUNARO SOFTWARE (CA)
PO Box 78, Macclesfield, Cheshire SK10 3PF

DISCOUNT SOFTWARE

GAMES		BUSINESS & TUTORIAL	
FOOTBALL MANAGER	5.95	GRAPHICS DESIGNER	9.25
CHESS	6.75	GUIDE TO BASIC 1	15.95
FLIGHT PATH 737	5.95	GUIDE TO BASIC 2	15.95
STEVE DAVIS SNOOKER	5.95	AMSWORD (EASY)	8.25
HUNTER KILLER	6.50	AMSWORD (ADVANCED)	15.95
SULTANS MAZE	6.25	AMSCALC SPREADSHEET	15.75
HOUSE OF USHER	5.45	HOME BUDGET	15.75
GEMS OF STRADOS	6.50	AMSCALC+	26.95
CODE NAME MAT	6.50	DEVPAC ASS/DISS	19.95
HARRIER ATTACK	6.50	ABERSOFT (FORTH)	19.95
SPANNER MAN	6.30	CONCISE BASIC SPEC	15.95
JET SET WILLY	6.75	CONCISE FIRMWARE SPEC	15.95
POLE POSITION	7.50	INTRO TO SOUND PT. 1	8.99

ALL PRICES INCLUDE POSTAGE & VAT
Please send cheques/postal orders to:
MJC SUPPLIES
'SCOJA' LONDON ROAD, HITCHIN, HERTS.

Amstrad CPC464 Speech Synthesizer

The dk'tronics Amstrad speech synthesizer and powerful stereo amplifier uses the popular SLO/256 speech chip and has an almost infinite vocabulary. It is supplied with a text to speech converter for ease of speech output creation. Everything you wish to be spoken is entered in normal English, without special control codes or characters, it is therefore extremely easy to use. The voicing of the words is completely user transparent and the computer can carry on its normal running of a program while the speech chip is talking. The speech output from SLO/256 is mono and directed to both speakers.

Stereo Output

To utilise the Amstrad stereo output on the back of the computer, the interface has a built in stereo amplifier, this gives all sound output a totally new dimension and greatly improves the sound quality and volume over the computer's internal speaker. Any sound that previously came out of the mono speaker will now be sent out via the interface in stereo. All programs that use the sound in anyway (i.e. commercial software) will now output through the interface, which is fitted with volume and balance controls.

Speech Synthesis

The Amstrad speech synthesis utilises parts of the spoken word known as allophones. These are actual sounds that go to make up speech. The SP0256 allophone speech synthesis technique provides the ability to synthesize an almost unlimited vocabulary. Fifty-nine discrete speech sounds (allophones) and five pauses are stored in the speech chip's internal rom.

Text to Speech

Although there are only 26 letters in the alphabet, letters have a totally different sound when used in different words. For example, The ''a'' in 'Hay' is much longer and softer than in 'Hat'. When you speak you automatically make adjustments because you know just how a word should sound Not quite so easy with a computer.

The machine code software is mainly developed to this mode of operation. 3.5K is used for tables which contain the rules & exceptions to the rules of the English Language.

e.g. I before E except after C) This therefore allows the user to enter words to be spoken in normal English.

Speakers

Supplied with the Speech Synthesizer are two high quality 4" speakers these have been designed to compliment theAmstrad Computer. They are fitted with 1 metre of cable and can be positioned for the best stereo effect. The synthesizer interface fits neatly on to the rear of the computer. It has a through connector to enable other interfaces (e.g. Disc Drive) to connect to the rear of the synthesizer for ease of expansion Please send S.A.E. for a copy of the instruction manual which will give full and comprehensive details.

New Basic Commands

There are 8 new Basic Commands which control all the functions of the interface. Making the Synthesizer very easy to use. You can even control the speed at which it will talk to you. Or use the synthesizer to create sound effects like a fourth sound channel.

10 PRINT '' 'AMSTRAD' ''

The above is an example of the Syntax for entering speech into the computer and shows how simple it is to use.

The instruction book gives comprehensive details and examples of how to use the interface both from machine code and basic.

How to Order

The Amstrad Speech Synthesizer costs only £39.95. You can obtain your synthesizer through any good computer store or by completing the order form and returning it to:
dk'tronics Limited, Shire Hill, Saffron Walden, Essex. OR by telephone quoting your barclaycard or access number. Orders normally despatched within 24 hours.

Please rush me

.....[QTY] Amstrad Speech Synthesizer at £39.95 +£1.25 p&p
I enclose cheque/PO/Cash for Total £.....
or debit my Access/Barclaycard No.

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

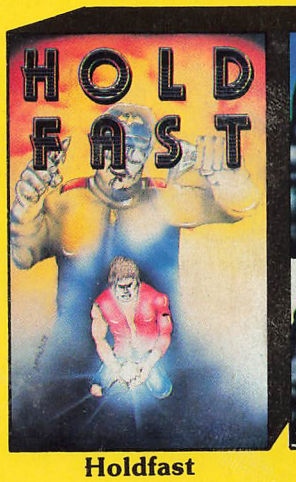
Signature.....
Name.....
Address.....

dk'tronics

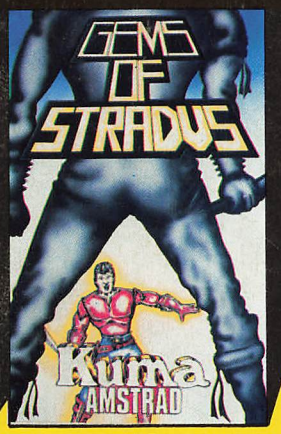
Saffron Walden, Essex CB11 3AQ
Tel: (0799) 26350 10 lines

the only choice

Kuma AMSTRAD CPC464 software

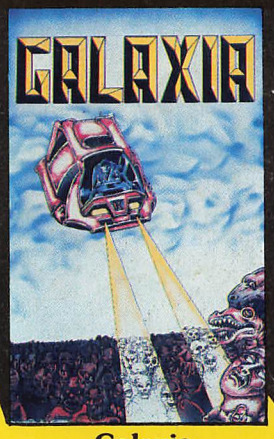


Holdfast



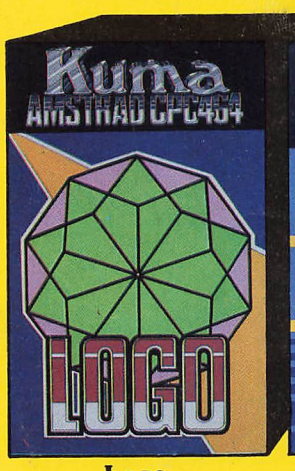
Gems of Stradus

Star Avengers

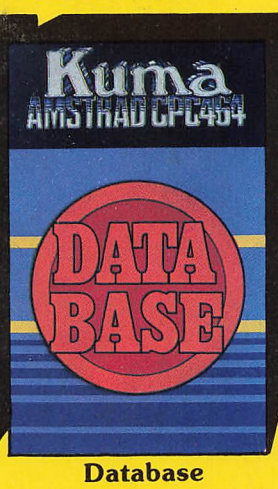


Galaxia

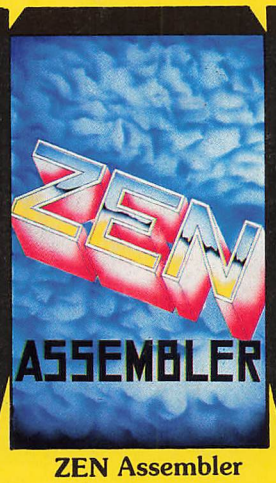
Music Composer



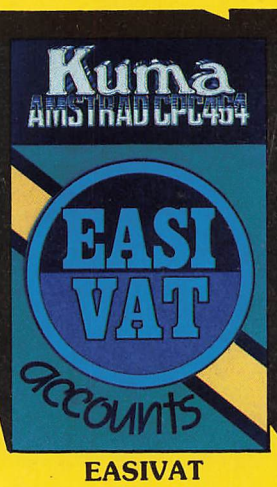
Logo



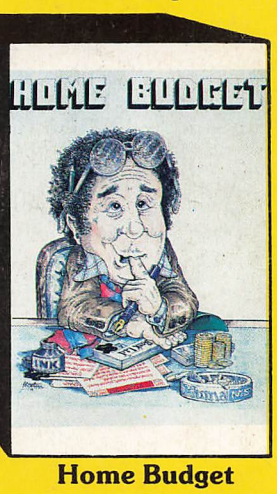
Database



ZEN Assembler

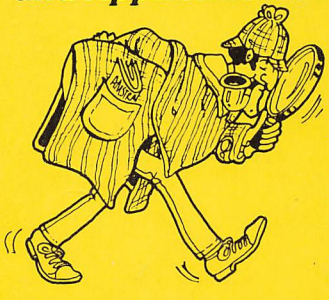


EASIVAT



Home Budget

An outstanding selection from Kuma's rapidly expanding range of Entertainment and Application Software for the Amstrad CPC 464 Micro-computer.



BOOK ● ***The Amstrad CPC 464 Explored.***

This superb book is designed to let every CPC 464 user, at whatever level, get the most from his computer. After an introductory section on the special Basic features, the book looks in depth at the excellent sound and graphic facilities.

Now available from selected branches of Co-op, Granada, LASKYS and John Menzies

Kuma Computers Ltd., Unit 12, Horseshoe Park, Horseshoe Road, Pangbourne, Berks RG8 7JW.

Please send full catalogue on Amstrad CPC464 products.

Name .....

Address .....

.....

..... Phone.....

I own an Amstrad CPC 464 computer ☐

Trade Enquiries Phone 07357-4335

Visitors wishing to call at our Pangbourne Manufacturing and Distribution Centre are advised to phone 07357-4335 first for an early appointment.